UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

Análisis y descripción del sistema embebido NVIDIA Jetson TX2 para el desarrollo de aplicaciones de inteligencia artificial

Autor/es

DANIELA ALVARADO JUSTINIANO

Director/es

JAVIER ESTEBAN VICUÑA MARTÍNEZ

Facultad

Escuela Técnica Superior de Ingeniería Industrial

Titulación

Grado en Ingeniería Electrónica Industrial y Automática

Departamento

INGENIERÍA ELÉCTRICA

Curso académico

2017-18

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INDUSTRIAL

TRABAJO DE FIN DE GRADO

TITULACIÓN: Grado en
Ingeniería Electrónica Industrial y Automática

CURSO: 2017/2018 **CONVOCATORIA: SEPTIEMBRE**

TÍTULO:

Análisis y descripción del sistema embebido Nvidia Jetson TX2 para el desarrollo de aplicaciones de inteligencia artificial

AUTOR: Daniela Alvarado Justiniano

DIRECTOR/ES: Javier Vicuña Martínez

DEPARTAMENTO: Ingeniería Eléctrica

Índice

Tabla de contenido

Memoria

Tabla de contenido

INDICE FIGURAS

INDICE TABLAS

Capítulo 1

Presentación

Este trabajo ha sido realizado por Dª Daniela Alvarado Justiniano, alumna del cuarto curso del Grado en Ingeniería Electrónica Industrial y Automática. Se presenta con el objeto de la obtención del título de Graduado en Ingeniería Industrial por la Universidad de La Rioja en el presente curso académico.

Objeto

El objetivo principal de este trabajo consiste en analizar las características y recursos hardware y software de la tarjeta NVIDIA Jetson TX2, así como experimentar y evaluar su potencialidad en el desarrollo de aplicaciones, entre otras, de reconocimiento de imagen mediante Deep Learning.

Alcance

Este trabajo contempla todas las fases necesarias para alcanzar los objetivos planteados en el apartado anterior. Como ya se ha dicho uno de los objetivos es el análisis hardware del sistema por lo que fue necesario la búsqueda de información de cada uno de sus componentes o módulos. Posteriormente toda la información encontrada fue analizada y estructurada para desarrollar de este modo un documento que facilite su comprensión. La información sobre las características de la tarjeta se tomó de la documentación proporcionada por NVIDIA, por lo que fue necesario registrarse como usuario para adquirirla. Al iniciar al análisis software se encontraron con conceptos como redes neuronales o Deep Learning, los cuales eran de gran interés por lo que se decidió analizarlos y estudiarlos en un bloque aparte. El conocimiento de estos conceptos facilito el desarrollo software, cuya información se obtuvo accediendo a diferentes plataformas oficiales, siendo estas las principales fuentes de información.

En la experimentación de los ejemplos fue necesaria la descarga de los diferentes algoritmos y su implementación en la tarjeta Jetson TX2. Una de las demos descargadas fue YOLOv3 (*You Only Look Once*) [69] quien presenta la incompatibilidad con la versión OpenCV 3.4. Por lo que se tuvo que desinstalar esta versión e instalar la versión 3.0. Además, se presentan dos programas realizados con OpenCV y el reentrenamiento de la red neuronal, que supuso la creación de bases de datos, la comprensión de sus parámetros y el análisis de los resultados. Con todo esto se abarca los requisitos necesarios para lograr alcanzar los objetivos planteados del trabajo.

Aunque se analizaron las diversas librerías que soporta la tarjeta Jetson TX2, no se llegaron a ejecutar o realizar ejemplos con todas ellas. Tampoco se llegó a crear una aplicación propia en concreto, ni interactuar con el procesamiento de audio, ya que los ejemplos llevados a cabo se centraron en el procesamiento de imágenes. No se profundizó con más detalle en el análisis hardware y software, ya que se consideró que las características planteadas eran las más importantes y trascendentales para este trabajo. En el reentrenamiento de la red neuronal no se llegó a realizar un análisis profundo que permitiese optimizar el algoritmo y analizar sus posibles variantes de modificación. Esto se debe a dos factores principalmente, el primero es que como el objetivo principal del trabajo no era este, no se le dio la máxima importancia, y se centró en el análisis del sistema. En el hipotético caso de haberlo llevado a cabo, este hubiera dado lugar a un planteamiento diferente del TFG (*Trabajo Fin de Grado*), ya que el desarrollo del algoritmo de la aplicación hubiera supuesto el cuerpo del trabajo. El segundo factor se debió al tiempo, como ya se ha comentado las posibles variantes del entrenamiento y la optimización de este supone tiempo con el cual no se contaba.

Justificación

La visión artificial es una disciplina que desde hace años viene desarrollándose y aumentando el número de aplicaciones en las que se utiliza. Todo esto ha sido posible gracias al desarrollo y avance de recursos hardware que soportan el procesamiento de grandes cantidades de datos. Siendo la visión artificial uno de los pilares en los que se basa la cuarta revolución industrial que se está produciendo y llevando a cabo durante estos últimos años. La esencia que caracteriza esta revolución es la adaptación de las cosas a las necesidades humanas, que se ha conseguido mediante la combinación de diversos factores como la hiperconectividad, el Big Data, la simulación, la fabricación aditiva, entre otras. Por lo que en la actualidad la gran mayoría de aplicaciones integran dispositivos capaces de almacenar grandes cantidades de datos en la nube. El procesamiento masivo de estos datos permite a las empresas ajustar su producción en función de la demanda de los clientes o realizar un aprendizaje profundo en base a ellos. La combinación de herramientas software y tecnologías basadas en estos factores han hecho que la visión que se tenía del mundo hasta ese momento diese un giro, enfocándose en nuevas expectativas. Una de ellas es la introducción de los llamados robots colaborativos en las industrias, los cuales se basan en el aprendizaje profundo y llevan incorporados un gran número de sensores y cámaras que les posibilitan trabajar junto al ser humano. Con la aparición de las impresoras 3D ha permitido la fabricación de piezas personalizadas, rápidas y de coste reducido, facilitando así la creación de prototipos propios a los cuales se les integra los algoritmos necesarios para llevar a cabo las respectivas aplicaciones. Dando lugar al creciente aumento del número de usuarios que desarrollan sus propias aplicaciones y las comparten con el resto de las personas, impulsando de este modo el aprendizaje de manera autónomo y experimental.

Por todo esto muchas empresas trabajan en el desarrollo y mejora de dispositivos que sean capaces de dar soporte a estas aplicaciones. NVIDIA es una de las compañías que más involucrada esta en este avance tecnológico, siendo una de las principales proveedoras de este mercado. Fue así cuando en 2017 saco a la luz la tarjeta Jetson TX2 para dar soporte aquellas

aplicaciones que requerían un bajo consumo, un tamaño reducido o el procesamiento de grandes cantidades de datos en un tiempo reducido. Esto último se consiguió gracias al procesamiento en paralelo de la GPU y CPU, ofreciendo de este modo un mayor rendimiento y mayor velocidad en el procesamiento de datos. La versión anterior TX1 que salió un año antes ya había sido incorporado en diversas aplicaciones y comercializadas como la Nintendo Switch, por lo que se espera que la nueva versión se integre en muchas más aplicaciones ya que presenta algunas mejoras.

Al disponer del material necesario en este caso de la tarjeta Jetson TX2, se consideró que era una buena oportunidad para introducirnos en el mundo de visión artificial, por lo que se decidió realizar este trabajo con el objetivo de realizar un análisis de las características y prestaciones que ofrece la SBC TX2. Para realizar un estudio lo más completo posible, además del análisis teórico se incluye la experimentación de algoritmos y demos que se realizaron para comprobar su funcionalidad. Un factor que facilitó llevar a cabo la ejecución de estos algoritmos es que las librearías y demos necesarias eran de código abierto por lo que no había que pagar sus licencias para conseguirlas. Además, el conjunto de la tarjeta incorpora todos los componentes mínimos necesarios para llevar a cabo su funcionamiento, algunos de ellos son el disipador de calor, antenas para la conexión Wifi o la cámara. Al contar con este último dispositivo ya incorporado no fue necesario la incorporación de otra cámara, ya que las pruebas se llevaron a cabo con la cámara de la tarjeta Jetson TX2.

Fases del TFG

Para la consecución de los objetivos planteados en el trabajo, los cuales se establecieron en el segundo apartado de este capítulo, se han completado una serie de etapas en los que se establecieron objetivos parciales.

- La primera fase supuso la documentación, búsqueda e interpretación de toda aquella información referida a los diferentes módulos y partes hardware del sistema.
- A medida que se iba encontrando esta información era necesaria su comprensión, análisis y razonamiento para plasmar el conjunto de todas estas ideas en el documento. Esto además supuso la búsqueda de conceptos desconocidos hasta el momento.
- Posteriormente se realizó la documentación de conceptos de primordial importancia como el Deep Learning o redes neuronales.
- Con el apartado anterior realizado, resulto más asequible abordar la parte software del sistema, ya que muchos de sus componentes hacen referencia a esos conceptos.
- Con el análisis teórico encaminado, se descargó e implementó el kit JetPack 3.1. Este lleva precargado un conjunto de librerías y ejemplos, con los cuales se empezó a realizar las pruebas de su funcionalidad. Estos fueron la detección de vehículos de un vídeo y los ejemplos de CUDA 8.0.

- El primer algoritmo que se descargó para llevar a cabo su ejecución fue Jetson-Inference, con el cual se realizó diversas pruebas de reconocimiento y detección de objetos, y segmentación de imágenes.
- Posteriormente se implementó el conjunto de tutoriales ofrecidos por Jetson-reinforcement, por lo que se ejecutaron un total de tres ejemplos en los que el aprendizaje se realiza interactuando con el entorno de la simulación.
- Después se descargó la versión de OpenCV 3.4 y se trabajó con un ejemplo que venía incorporado. Este consistía en filtrar la imagen capturada por la cámara y convertirla a diferentes formatos como escalas de grises y detección de contornos.
- El siguiente ejemplo que se abordó fue YOLOv3, por lo que se tuvo que implementar este algoritmo. En el momento de ejecutarlo se obtuvo el problema que YOLOv3 no es compatible con la versión 3.4 de OpenCV. Para solucionar este problema, se tuvo que desinstalar esta versión e instalar la 3.0.
- Una vez instalado OpenCV 3.0 se llevaron a cabo la realización de las pruebas con el algoritmo YOLOv3, con el que se detectaba e identificada los objetos presentes en la imagen.
- Se decidió realizar el reentrenamiento de la red neuronal, para lo cual fue necesario crear una base de datos con un conjunto de imágenes y se siguió el método utilizado por (Medium, 2018) [54] y (Redmon, 2018) [83].
- Se realizaron tres entrenamientos los cuales se efectuaron en el siguiente orden:
  - Entrenamiento de botellas de las marcas de bebidas Coca-Cola y Fanta.
  - Entrenamiento de latas de las correspondientes marcas anteriores.
  - Entrenamiento de los logos de estas dos marcas.
- Como fase final se desarrollaron dos programas basados en OpenCV, para lo cual fue necesario consultar diversas fuentes como foros y la página oficial de OpenCV.
  - Primero se desarrolló el programa que detecta objetos según el color establecido, en este caso la gama azul. La región detectada del objeto es limitada por un cuadrado y se indica sus coordenadas en dos dimensiones.
  - El segundo programa consiste en detectar aquellos objetos circulares de la imagen e indicar su radio, y si este supera un rango establecido o no. En ambos programas la imagen con la que se trabaja es la capturada por la cámara incorporada de Jetson TX2.

Capítulo 2

Introducción

2.1.Resumen

El trabajo que se presenta a continuación trata de una de las disciplinas que más está avanzando y progresando en la actualidad, que es la visión artificial. Debido a este gran auge diversas compañías trabajan en el desarrollo de módulos que den soporte a las aplicaciones que se están creando o estudiando, para llevar a cabo en un futuro no muy lejano. De este modo NVIDIA creó la tarjeta Jetson TX2 que es el dispositivo en el que se basa el desarrollo de este trabajo. En él se presenta el análisis de las características y prestaciones que ofrece la tarjeta, y se muestra la experimentación llevada a cabo en base a los ejemplos realizados. De esta forma se pretende comprobar su aplicabilidad mediante la ejecución de diferentes algoritmos.

Este TFG se inicia con una breve introducción de conceptos relacionados con la visión artificial y se presenta la compañía NVIDIA, que es la creadora de esta tarjeta. Posteriormente se muestra el análisis teórico en el que se describe los recursos hardware y software del sistema. En el estudio hardware se analiza cada uno de sus componentes, y se trata de una de sus principales características que es el funcionamiento paralelo de la GPU y CPU.

Antes de continuar con los recursos software se presenta un capítulo en el que se tratan conceptos cuyo conocimiento facilitará continuar con el desarrollo del trabajo, ya que se hará referencia a estos en los siguientes capítulos. Tras esto se continua con el análisis software en el que se presenta su SO (*Sistema Operativo*) Linux, Ubuntu y el kit Jetpack 3.1 en el que se describe las diferentes plataformas o librerías con las que es compatible Jetson TX2.

Para completar el análisis y comprobar su funcionalidad se realizaron varios ejemplos desde algunos que ya estaban incluidos en el Jetpack 3.1, hasta otras demos que se descargaron posteriormente. Además, se muestran dos pequeños programas realizados con OpenCV y el reentrenamiento de las redes neuronales que se llevó a cabo mediante YOLOv3 para detectar dos marcas de refrescos. Este entrenamiento se realizó de tres modos, en el primero se pretende que detecte las botellas de estas dos marcas. El segundo se basa en la detección de latas, y con el último se pretende que detecte todos los objetos que contengan los logos de estas dos marcas así sean botellas, latas, camisetas o tapas.

2.2.Abstract

The project shown next is based on a field of expertise that nowadays is making a greater progress, computer vision. Due to the increase in the demand for computer vision applications, companies are developing new products to be applied to new real-life applications in the near future. Therefore, NVIDIA developed the module Jetson TX2. This SBC was used on this project. An analysis of its characteristics and capabilities was performed executing a great variety of examples. Also, different algorithms were executed to test the capabilities of the module.

The dissertation starts showing a small introduction to the topic and the company that designed the board, NVIDIA. Then, a theoretical approach was followed in order to describe its hardware and software. All of the components of the board and its most important characteristic, parallel computing, were studied in the hardware chapter. Before starting the software chapter, a brief introduction to new concepts was written in order to be easier for the reader to follow and understand the following topics of the dissertation. In the software analysis, the OS, Ubuntu, the Jetpack 3.1 and the compatible software libraries are studied.

To finalise the analysis. different examples were executed. These examples were downloaded from the internet, nevertheless, some of them were available in the Jetpack3.1. Two examples of OpenCV were coded and executed. Also, a custom neural network training to distinguish products between two fizzy drinks companies using the software YOLO V3 was accomplished. Three approaches were followed. Firstly, the training was based on the detection of the bottles of both companies. Then, cans from both companies were trained instead. On the last attempt, the neural network was trained to detect the logos of the brands, so, any product with that logo can be identified.

2.3.Introducción

En el presente TFG se muestra el análisis de las características y recursos hardware y software que ofrece la tarjeta Jetson TX2. En el estudio hardware se describe y analiza cada uno de sus componentes, mientras que en el software se presenta su SO y las diversas librerías y plataformas con las que es compatible.

Además, se llevará a cabo la experimentación de ejemplos mediante el uso de la tarjeta Jetson TX2, y se realizarán dos pequeños programas con la librería OpenCV. Por último, se muestran los resultados obtenidos del reentrenamiento de la red neuronal realizado con el algoritmo YOLOv3. La realización de este documento se apoya en diversas plataformas, tutoriales, foros y documentación, lo cual ha sido posible gracias al creciente número de personas que han aumentado y mejorado todas estas fuentes de información.

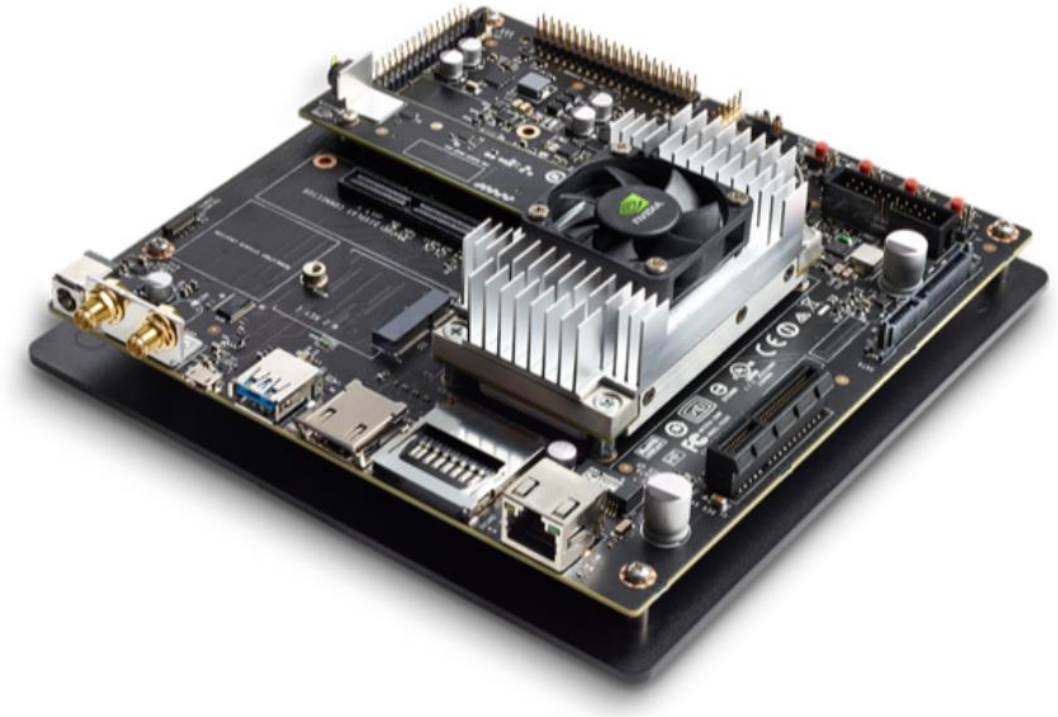


Figura 1: SBC Jetson TX2

El funcionamiento y prestaciones que ofrece la tarjeta Jetson TX2 son similares a las de un ordenador, pero a tamaño reducido ya que integra todos sus componentes en una placa. Por ello es considera una SBC (*Single Board Computer*) que lleva incorporada una cámara, un disipador de calor, antenas para la conexión Wifi y un conjunto de elementos que hacen posible la conexión con dispositivos periféricos del sistema. El módulo Jetson TX2 que es el elemento principal de la tarjeta presenta las características idóneas para ser incorporado en aplicaciones de potencia y tamaño reducido, así como en robots, drones o cámaras de seguridad.

2.4. “Estado del arte”

Tan sólo han pasado cincuenta años y hoy en día se convive con tecnología que en esos momentos parecían de ciencia ficción. Los robots, los hologramas, videojuegos virtuales o los coches autónomos eran solo especulaciones, pero estos se están convirtiendo en realidad y van en aumento. Gran parte de los avances tecnológicos que han tenido lugar en las últimas décadas se han realizado mediante Deep Learning, el cual se basa en el Big Data y en las diversas arquitecturas de redes neuronales existentes. Aunque las investigaciones y modelos basados en Deep Learning llevan décadas desarrollándose, estos no se habían llevado a cabo debido a la inexistencia de un soporte hardware capaz de procesar tanta información. A continuación, se presentan estos conceptos y se muestra cómo se han visto influenciados por el avance tecnológico.

En la actualidad prácticamente todas las cosas están conectadas a internet lo que ha dado lugar al aumento masivo de datos. Cuando el volumen de estos datos supera los 30 o 50 TB, presentan una estructura compleja y resultan difíciles de capturar debido a su velocidad, se les denomina Big Data (PowerData,2018) [80]. Están presentes en todos los sectores, ya que en base a ellos las organizaciones disponen de una gran cantidad de datos sobre sus clientes. Esto ayuda a las empresas a realizar movimientos menos arriesgados y a obtener mayores ventas y beneficios, ya que en base al análisis de estos datos conocen de antemano las experiencias, valoraciones, gustos y tendencia entre los clientes. Tres de los avances tecnológicos que revolucionaron el Big Data fue la creación de Internet, la introducción de Google como un sistema de búsqueda y el aumento del uso de móviles. (IGN-Soluciones de gestión para pymes,2018) [37]

La característica de volatilidad que presenta el Big Data y la masiva cantidad de datos impedía su análisis, ya que los datos cambian muy rápidamente lo que hace que tenga una validez muy corta. Por lo que era necesario el desarrollo de un soporte hardware potente, lo cual fue posible con la mejora de las GPUs. Siendo estas capaces de procesar grandes cantidades de datos de forma paralela, reduciendo así el tiempo de procesamiento y evitando los cuellos de botellas. Cuando se empezó a utilizar la GPU en el procesamiento de datos se le denomino inicialmente GPGPU, pero al tiempo cayó en desuso ya que se consideró que tanto el procesamiento de gráficos como el de datos requerían de los mismos requisitos para el procesamiento de coma flotante, por lo que el mismo término hace referencia a ambos.

Uno de los principales desarrolladores de estas unidades es NVIDIA, que ha llevado a cabo el desarrollo de cinco arquitecturas en la última década. Estas GPUs son utilizadas principalmente para el procesamiento y cálculo de grandes volúmenes de datos, y en aplicaciones de procesamiento de imágenes. A continuación, se presentan las principales características de las arquitecturas desarrolladas por NVIDIA y en anexos se muestra una explicación de los shaders y la distribución que presentan las siguientes arquitecturas.

- Tesla: Esta microarquitectura revoluciono el mundo tecnológico con la introducción de shaders unificados [51]. Su controlador es compatible con Direct3D 10 [16] ya que presenta el modelo shaders 4.0 [52] y OpenGL 2.1 [45]. La primera tarjeta gráfica en introducir estas características fue GeForce 8 cuyos SPs (*Stream Processors*) [56] presentan una distribución en hilos por lo que resultan más fáciles de construir y pueden funcionar a diferentes velocidades de reloj. Los anteriores SPs eran shaders que realizaban

funciones específicas, cuyos bloques no estaban unidos por hilos, aunque podían realizar más de una operación, pero su rendimiento dependía de la mezcla y orden de las instrucciones. (En.wikipedia. org, 2018) [22]

- Fermi: Su principal objetivo fue aumentar el rendimiento en la computación y teselación de imágenes. Esta arquitectura a diferencia de la anterior cada SM (*Streaming Multiprocessos*) [53] está compuesto por 32 núcleos CUDA [13], y no por 8 como era en el caso de Tesla. Introduce el estándar de coma flotante, en el que fusiona las operaciones de suma y multiplicación *FMA*. Ofreciendo así una doble precisión, ya que la ALU (*Unidad Aritmética Lógica*) [1] admite 32 bits por instrucción. (NVIDIA's Next Generation CUDA Compute Architecture: Fermi, 2009) [71]

- Kepler: Se caracteriza por ser la primera en centrarse en la eficiencia energética, la cual se obtuvo mediante la implementación de un reloj GPU unificado. Para hacer que toda la GPU funcione a la misma velocidad de reloj y siga ofreciendo altos rendimientos, se tuvo que implementar núcleos adicionales. Esto se obtuvo mediante el cambio de SM a SMX [54], la incorporación del paralelismo dinámico y la arquitectura Hyper-Q [34].

- Maxwell: Esta arquitectura remodificó el diseño de los SM con la presentación de los SMM, ofreciendo una mayor eficiencia que los SMX de Kepler. Se divide en dos generaciones, en la primera se presentó las modificaciones realizadas en el diseño general de los SMM. En la que su arquitectura a diferencia de los SMX se divide en cuatro cuadrantes y cada uno contiene 32 SP, los cuales a la vez contienen 32 núcleos CUDA, haciendo un total de 4096 núcleos. La segunda generación se caracteriza por introducir mejoras tecnológicas como dar soporte HDMI 2.0 (High Definition Multimedia Interface) o DSR (Dynamic Super Resolution). (Harris et al., 2018) [35]

- Pascal: Su principal característica es su compatibilidad con la memoria HBM2 (*High Bandwith Memory*) [31]. Por lo que es capaz de realizar el procesamiento de datos a mayor ancho de banda, mejorando la eficiencia y el rendimiento computacional. Con la incorporación de HBM2 los bloques de memoria se disponen en dos bloques verticales unidos mediante matrices a los SM, en vez de estar colocados en torno a la GPU como sucedía con GDDR5 [27] (*Graphics Double Data Rate*). De este modo el área que se requiere es menor y la energía para la transferencia de datos es ve reducida.

- Volta: Es la arquitectura más reciente de GPUs que ha presentado NVIDIA, la cual está orientada para dar soporte a las aplicaciones basadas en Deep Learning. Presenta una nueva arquitectura SM, en la que incorpora los núcleos tensors, un aumento de frecuencia de reloj y mayor ancho de banda. (NVIDIA TESLA V100 GPU ARCHITECTURE, 2017) [77]

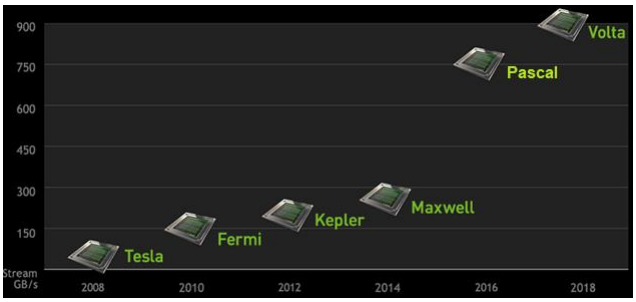


Figura 2: Arquitecturas GPUs NVIDIDA

En la tabla inferior se muestra un modelo de ejemplo de cada arquitectura y sus principales características. (En.wikipedia.org, 2018) [25] (Nvidia.es, 2018) [73]

	Tesla (C1060)	Fermi (M2090)	Kepler (K80)	Maxwell (M60)	Pascal (P100)	Volta (V100)
Año	2008	2010	2014	2015	2016	2017
Chip	GT200	GF110	GK210	GM204	GP100	GV100
Nº GPUs	1	1	2	2	1	1
M. Banda Ancha (T.)	102.4 GB/s	177.6 GB/s	480 GB/s	320 GB/s	732 GB/s	900 GB/s
Memoria (T.)	GDDR3 [26] 4 GB	GDDR5 6 GB	GDDR5 24 GB	GDDR5 16 GB	HBM2 16 GB	HBM2 16 GB
Nº Núcleos CUDA	240	512	4992	4096	3584	5120

Tabla 1: Arquitecturas GPUs NVIDIA

En 1943 McCulloch y Pitts propusieron el primer modelo en el que las neuronas eran representadas mediante un modelo matemático. En los años posteriores se continuaron realizando estudios y en 1969 se plantea el algoritmo de Retropropagación.

Aunque se estaban haciendo importantes avances en el análisis y planteamiento de modelos basados en redes neuronales, los sistemas hardware disponibles no eran lo suficientemente potentes para llevar a cabo la implementación de grandes redes en los dispositivos. Esto se debe a que la red realiza cálculos muy intensivos, derivadas parciales y funciones como la sigmoide durante el entrenamiento, por lo que al no disponer de dispositivos hardware potentes la implementación del número de neuronas y capas era limitada. Por tanto, las aplicaciones que se realizaban se limitaban a redes relativamente pequeñas. Esto cambió con la mejora de potentes GPUs con las que el número de neuronas y capas crecieron permitiendo el desarrollo de aplicaciones basadas en Deep Learning.

A nivel de desarrollo hardware la compañía NVIDIA es una de las principales protagonistas que ha intervenido en el avance de las redes neuronales, mientras que a nivel software ha sido Google. Quien dio a conocer en 2011 DistBelief que es un entorno que utiliza núcleos de máquinas para el entrenamiento de redes neuronales. El algoritmo de entrenamiento que utiliza es *Downpour SGD* que divide los datos de entrada en bloques, y cada uno de estos bloques tiene una copia del modelo de la red de entrenamiento. Esta red a su vez es dividida en modelos pequeños que serán entrenados por distintas máquinas, por lo que la carga de trabajo es distribuida entre todos los núcleos utilizados.

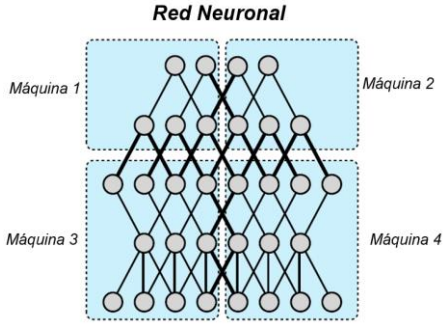


Figura 3: Red dividida en modelos pequeños

Se observa que las uniones entre las neuronas pueden estar dentro de una misma máquina, en cuyo caso la información es procesada por los núcleos de esta, o pueden unir dos neuronas de diferentes máquinas, por lo que en este caso se produce una transferencia de información entre las máquinas.

Todas las réplicas o copias de la red se comunican con un mismo servidor, el cual esta divido en fragmentos donde se acumulan los resultados transferidos. Este modelo es asíncrono por lo que la actualización de las réplicas de los modelos de la red se ejecuta de manera independiente, al igual que la actualización de los datos en los distintos fragmentos del servidor. Por lo que, ante un fallo las otras réplicas del modelo continúan procesando sus datos de entrenamiento y actualizando los parámetros en el servidor.

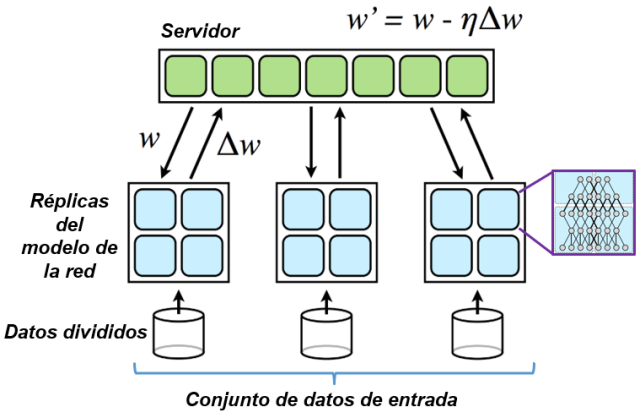


Figura 4: Downpour SGD

Al iniciar su funcionamiento cada réplica del modelo de la red solicita a su respectivo fragmento del servidor los datos de los parámetros de su modelo. Este le transfiere estos datos, por lo que cada réplica procesa estos datos determinando su gradiente, el cual es enviado al servidor para determinar el valor actual de sus parámetros. (Large Scale Distributed Deep Networks,n.d) [47]

En 2012 Google creo una red neuronal de 17000 parámetros, cuyo objetivo era reconocer gatos y para ello necesito 16000 núcleos de CPU (Nvidia.es, 2018) [74]. Continúo mejorando DistBelief y en 2015 creó TensorFlow, que es una biblioteca de código abierto en la que se trabaja con conjuntos de datos multidimensionales, denominados Tensor. TensorFlow se convirtió en la herramienta más utilizada por los desarrolladores, por lo que las empresas continuaron optimizando sus productos para darle soporte y es así como NVIDIA ha dado a conocer su nueva arquitectura Volta. La cual se caracteriza por la introducción de núcleos Tensor de precisión mixta, con los que se puede realizar operaciones matriciales, destinado para aplicaciones basadas en Deep Learning. Antes de que NVIDIA diera a conocer su nueva arquitectura Volta, Google en 2016 ya había desarrollado su TPU (*Tensor Processing Unit*) [63] destinada para el aprendizaje automático.

2.5.Bloques del TFG

La información que presenta este trabajo se podría dividir en cuarto grandes bloques dentro de los cuales se encuentran los distintos apartados que se han analizado, estudiado, ejecutado y desarrollado a lo largo del TFG.

La estructura que presenta es la siguiente:

- Primer bloque: Descrito en el capítulo 3

En él se realiza una breve introducción de la tecnología desarrollada por NVIDIA y las diferencias que existen entre la versión de Jetson TX2 y la anterior TX1, también se muestra una aplicación real que saldrá a la venta este año.

- Segundo bloque: Descrito en el capítulo 4

En este bloque se presenta el cuerpo principal del trabajo, ya que está compuesto por al análisis teórico del sistema. En él se analizan las siguientes partes:

- Recursos hardware: se describen y analizan cada uno de sus componentes tres de los más importantes son el módulo Jetson TX2, el disipador de calor y la cámara que lleva incorporada.
- Conceptos relevantes: se realiza un estudio más exhaustivo de conceptos como redes neuronales, Deep Learning y el procesamiento de imágenes.
- Recursos software: en él se presenta su SO, Ubuntu y el kit JetPack 3.1 en el que se describe las diferentes plataformas, librerías o algoritmos que integra.

- Tercer bloque: Descrito en los capítulos 5 y 6

Este bloque está compuesto por el conjunto de todos los programas ejecutados que se han llevado a cabo.

- Ejemplos incorporados de CUDA 8.0.
- Jetson-Inference.
- Tutoriales de Jetson-reinforcement.
- OpenCV con la realización de los dos programas y el ejemplo de filtros ejecutado.
- YOLOv3
- Reentrenamiento de la red neuronal realizado de los tres casos: botellas, latas y logos.

- Cuarto bloque: Descrito en el capítulo 7

En este bloque se plantean las conclusiones obtenidas una vez realizado el análisis teórico y la experimentación llevada a cabo mediante la ejecución de los diversos algoritmos.

Capítulo 3

NVIDIA Jetson TX2

3.1.Introducción

La empresa NVIDIA se caracteriza por desarrollar unidades de procesamiento gráfico y circuitos integrados para el desarrollo de diversas aplicaciones de tiempo real, de visión artificial, de dispositivos móviles o para juegos de ordenadores. Aunque NVIDIA es una de las principales proveedoras del mercado tecnológico, desde 2014 se ha centrado en cuatro bloques que son los videojuegos, centros de datos, el avance de dispositivos autónomos y la visualización computacional. En el desarrollo de este último se llevó a cabo la creación de la SBC TX2 que fue lanzada al mercado en el 2017. El paquete que NVIDIA ofrece se muestra en la imagen inferior e incluye lo siguiente:

- El SBC TX2, que lleva incorporado una cámara y el disipador de calor con el ventilador.
- Una pequeña guía de seguridad del kit de desarrollo Jetson TX2.
- Un pequeño folleto con una breve introducción y con sus principales características.
- Dos antenas para la conexión Wifi.
- Una fuente de alimentación de 19 V y 4.7 A. Su potencia total disponible es de 90 W, de los cuales 15 W son los que consume el módulo Jetson TX2, quedando disponible el resto para la conexión de cámaras u otros periféricos.
- Un cable negro USB con terminal micro 2.0 y 3.0.
- Un cable blanco USB con terminal micro 2.0 y 3.0.

Figura 5: Componentes del paquete Jetson TX2

Las características y dimensiones que presenta están orientadas a aplicaciones de inteligencia artificial, como puede ser el control de drones, robots, cámaras inteligentes, sistema de seguridad e incluso en coches autónomos, que es en lo que se está trabando en muchas empresas actualmente. Un ejemplo de una aplicación real son las gafas de realidad mixta "Magic Leap One" que saldrán a la venta a finales de este año según (Lang, 2018) [46] y (Álvarez, 2018) [4]. La empresa Magic Leap incorpora en el sistema de estas gafas el módulo Jetson TX2, permitiéndole un procesamiento paralelo de la CPU y GPU. Aunque aún no se conoce con exactitud su aspecto físico, la demo que se ha presentado de cómo será físicamente es la siguiente.

Figura 6: Magic Leap One

En la demo que se presentó aparece una criatura que lanza rocas y se muestra que las manos interaccionan con el sistema mediante la detección de los gestos. Pero las expectativas son mayores, ya que se espera que sea capaz de realizar seguimiento ocular, rastreo de manos y la ubicación en tiempo real.

Una de las grandes ventajas que ofrece el SBC TX2 es que es de código abierto, por lo que se pueden descargar los módulos o demos desde la plataforma Github, para posteriormente implementarlo en la tarjeta. Como se ha dicho anteriormente NVIDIA proporciona documentación y videos en su página web, y además en YouTube hay varios tutoriales en los que explican como descargar algunos módulos y se muestra su funcionamiento.

3.2. Jetson TX2 vs TX1

La versión anterior presentada por NVIDIA fue la Jetson TX1 que salió al mercado en 2016 y está basada en la arquitectura Maxwell. La tarjeta TX2 se basa en la arquitectura Pascal y presenta las mismas capacidades de la TX1 y algunas mejoras y características adicionales. A continuación, se muestran las mejoras que presenta la arquitectura Pascal a diferencia de Maxwell.

- Transmisión de datos de manera más simplificada y rápida.
- Nueva arquitectura del entorno SM.
- Mejoras en la programación y comandos de instrucciones.
- Nuevas operaciones aritméticas.
- Mejoras en el soporte de grandes espacios de direccionamiento.
- Mayor eficiencia energética, permitiendo un adecuado rendimiento en dispositivos de potencia limitada.

El kit de desarrollo de ambas tarjetas presenta los mismos módulos, el aspecto de ambas es muy similar y el paquete que NVIDIA proporciona incorpora los mismos componentes. Algunas de las diferencias de estas dos versiones se observan en la siguiente tabla que aparece en el enlace (NVIDIA, 2018) [75], en la que se muestran las especificaciones técnicas de los módulos TX1 y TX2.

	Jetson TX1	**Jetson TX2**
Gráficos	Arquitectura Maxwell: 256 Núcleos CUDA	Arquitectura Pascal: 256 Núcleos CUDA
CPU	ARM Cortex-A57, de núcleo cuádruple	ARM Cortex-A57, de núcleo cuádruple y Denver 2, de núcleo doble
Vídeo	Codificación 4kx2k a 30 Hz Descodificación 4Kx2K a 60 Hz (10 bits)	Codificación 4kx2k a 60 Hz Descodificación 4Kx2K a 60 Hz (12 bits)
Memoria	LPDDR4 de 4 GB y 64 bits 25,6 GB/s	LPDDR4 de 8 GB y 128 bits 25,6 GB/s
Pantalla	2 interfaces DSI, DP 1.2/HDMI/ Edp 1.4	2 interfaces DSI, 2DPI 1.2/HDMI 2.0/ Edp 1.4
CSI	Hasta 6 cámaras/ CSI2 D-PHY 1.1 (1.5 Gbps/vía)	Hasta 6 cámaras/ CSI2 D-PHY 1.2 (2.5 Gbps/vía)
PCIe	Gen 2 l 1x4 + 1x1	Gen 2 l 1x4 + 1x1 o 2x1 + 1x2
Almac. de datos	SDIO, SATA de 16 GB	SDIO, SATA de 32GB
Otros	UART, SPI, I2C, GPIOs	UART, SPI, I2C, GPIOs, CAN
USB	USB 3.0 y USB 2.0	
Conectividad	1 GB Ethernet, 802.11 ac WLAN, Bluetooth	

Tabla 2: Especificaciones técnicas TX1 y TX2

Capítulo 4

Análisis Teórico

En el análisis teórico se desarrollan los estudios hardware y software del sistema. Primero se presenta su parte hardware en el que se trata el procesamiento paralelo de la CPU y GPU, además se describen todos sus componentes que permiten y limitan la conexión con los diversos periféricos y el estudio térmico de la placa. Posteriormente se presentan algunos conceptos de interés relacionados con la visión artificial y en el último bloque se muestra el estudio software en el que se describe y analiza su SO y los diversos módulos que incorpora el Jetpack 3.1. (Elinux.org,2018) [15]

4.1. Análisis Hardware

Para iniciar el estudio de la SBC TX2 se realiza un análisis hardware en el que se describen sus diferentes componentes, cuya información se ha tomado como base de referencia de los PDFs descargados de NVIDIA (NVIDIA Jetson TX1/TX2 Developer Kit Carrier Board, 2017) [69] y (NVIDIA Jetson TX2 System-on-Module Pascal GPU + ARMv8 + 8GB LPDDR4 + 32GB eMMC + WLAN/BT, 2017) [70].

Además, se habla de la arquitectura CUDA de sus núcleos y del funcionamiento de su memoria. En las imágenes inferiores se muestran todos estos módulos o componentes y la enumeración en la que son descritos a continuación.

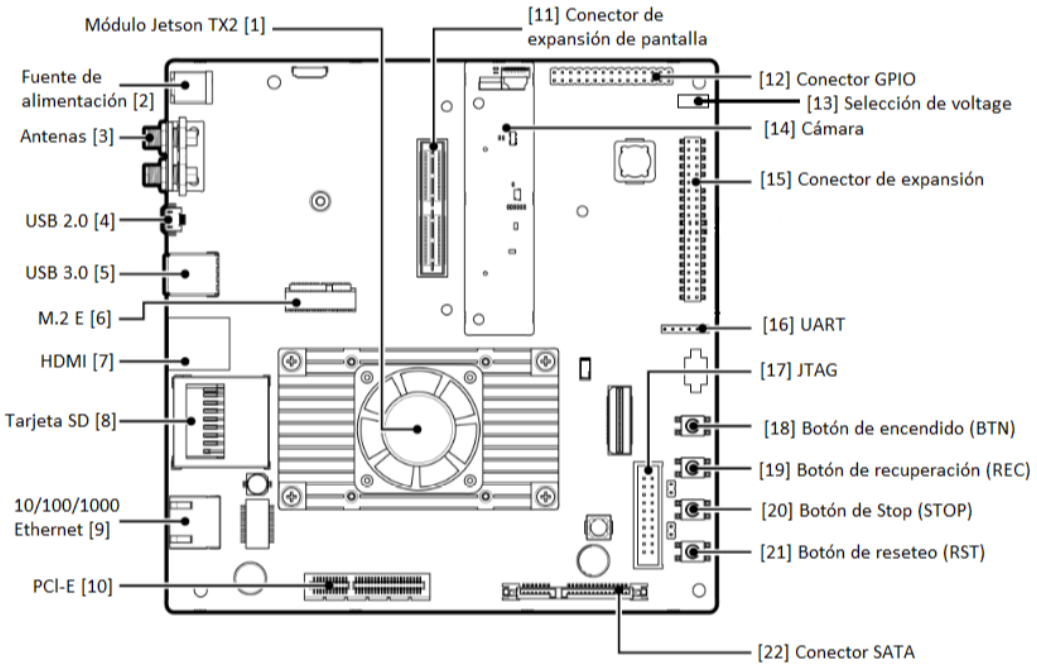


Figura 7: Diagrama general de la tarjeta de soporte y desarrollo para el SBC TX2

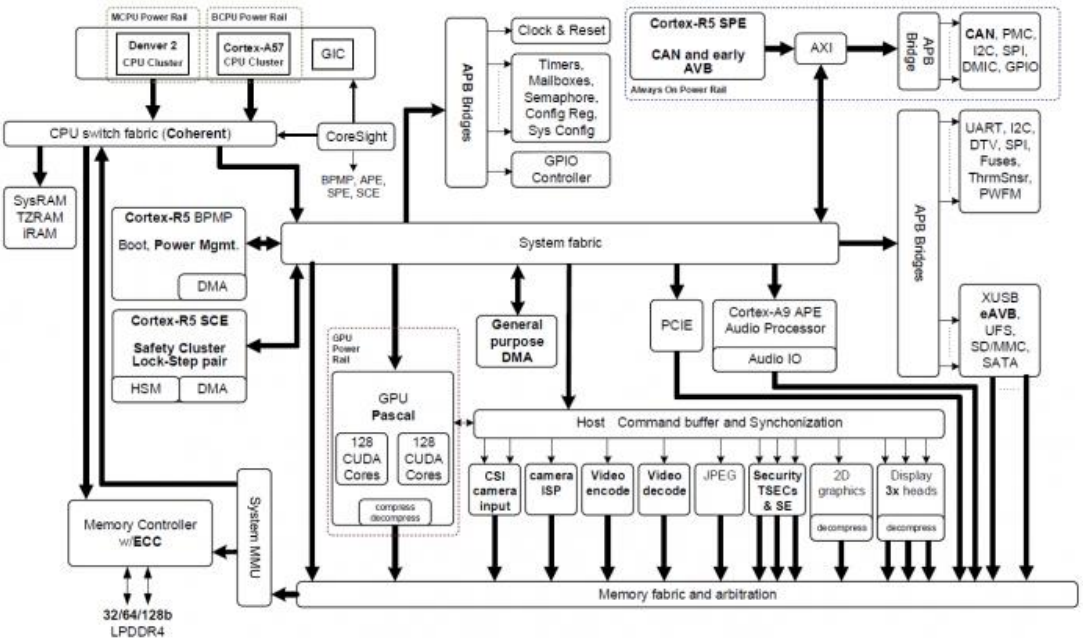


Figura 8: Diagrama de flujo del sistema

4.1.1 Módulo Jetson TX2 (SBC)

Es el componente principal de la placa, ya que integra los elementos esenciales para su funcionamiento como es la GPU y CPU, entre otros. Este módulo está situado debajo de un pequeño sistema de disipación de calor, colocado para evitar el sobrecalentamiento del sistema. En la imagen inferior se muestra el aspecto que presenta este módulo.

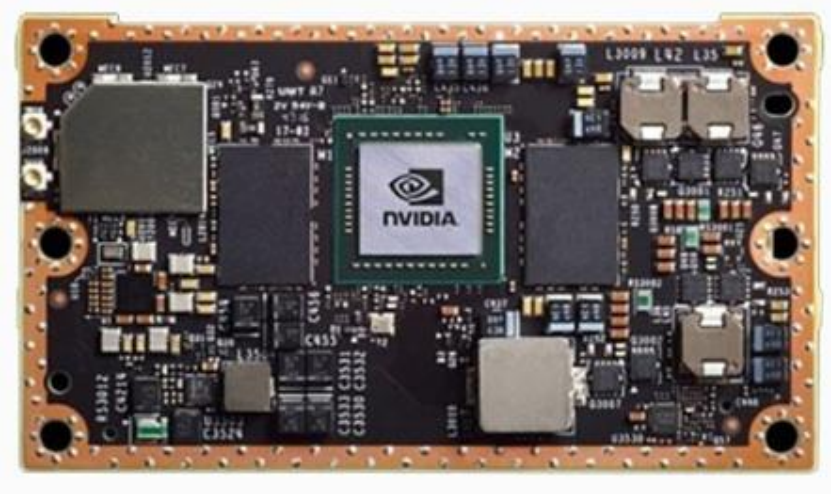


Figura 9: Módulo SBC

Los principales componentes integrados que incluye este módulo son los siguientes:

- Tegra X2, SoC (*System on Chip*) [55]: Es el sistema central de la placa, el cual está formado por la GPU y la CPU.
- Memoria LPDDR4 de 8 GB.
- Módulo eMMC 5.1 de 32 GB de almacenamiento.
- PMIC, reguladores, monitores de voltaje y potencia.
- Conectores de antena para conexiones WLAN y Bluetooth.
- Sensores de temperatura.

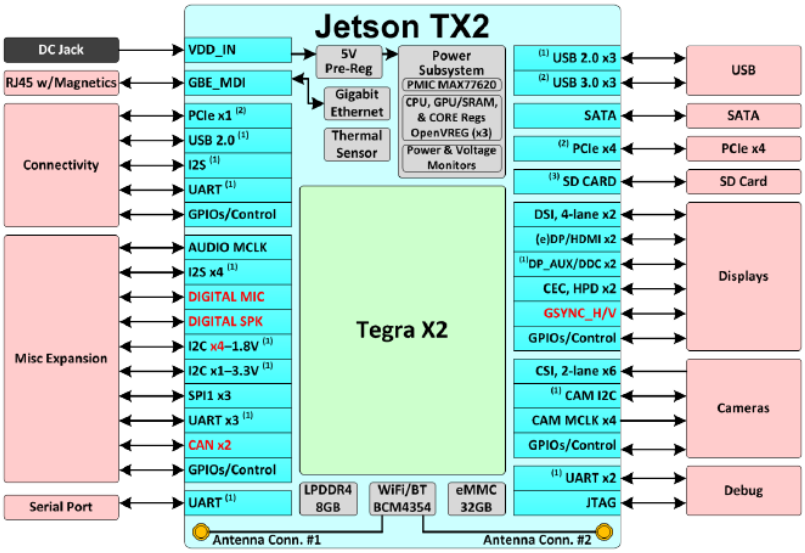


Figura 10: Conexión Jetson TX2

Esta tarjeta se caracteriza por el procesamiento paralelo de su GPU y CPU, por lo que cuando el módulo recibe un código, la GPU se encarga de ejecutar las funciones más complejas e intensas, mientras que la CPU se encarga de procesar el resto de las funciones. Ambas ejecutan su parte de código de manera independiente y a la salida los resultados son unificados, obteniendo de este modo una única salida de datos.

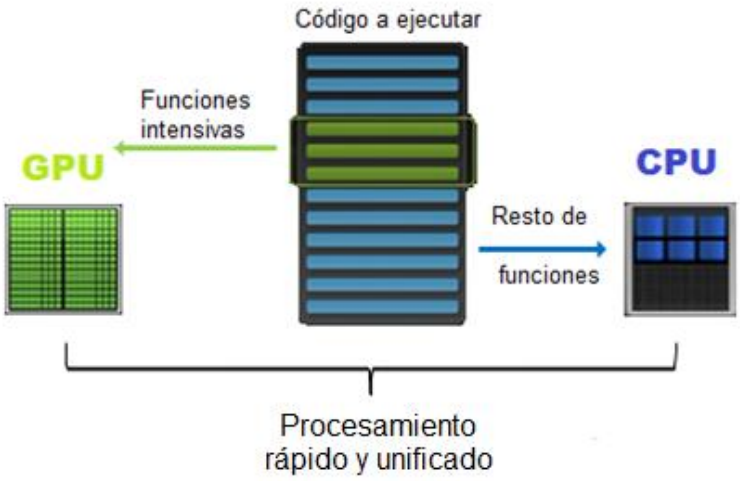


Figura 11: Procesamiento CPU y GPU

GPU

Es la unidad de procesamiento gráfico y la encargada de realizar las operaciones de coma flotante, disminuyendo de este modo la carga de trabajo de la CPU. Se basa en la arquitectura Pascal y está compuesta por 256 núcleos CUDA (Nvidia.es, 2018) [75] [76]. No posee memoria propia, sino que comparte 8 GB con la RAM (*Random Acces Memory*) del sistema y es compatible con lenguajes de alto nivel como C y C++, y de bajo nivel como Python.

La principal diferencia entre esta tarjeta basada en la arquitectura CUDA y las de versiones anteriores, es que estas últimas necesitaban dos tipos de procesadores para su funcionamiento. Un procesador era de vértices y otro de fragmentos, cada uno se dedicaba a realizar distintas tareas y con repertorios de instrucciones diferentes, por lo que los procesadores tenían diferentes cargas de trabajo. Sin embargo, la arquitectura CUDA es unificada, en la que los núcleos son procesados como un conjunto de hilos que utilizan el mismo repertorio de instrucciones y de recursos. Este módulo SBC tiene 256 núcleos CUDA o SP, que están divididos en grupos de 32 formando un SM. En total hay 8 SM que están conectados entre sí por la memoria común, a la que tienen acceso todos los hilos. (INTRODUCCIÓN A LA PROGRAMACIÓN EN CUDA, 2016) [39]

El conjunto de 32 hilos que unen los núcleos de un SM recibe el nombre de warp y se ejecutan en paralelo, por lo que todos comienzan con la misma instrucción, aunque posteriormente se bifurcan y se ejecutan independientemente. Cada SM tiene una memoria compartida interna lo que hace una comunicación entre hilos mucho más rápida y a su vez cada hilo posee una memoria local. Todo esto hace que no sea necesario la transferencia de datos con otras memorias de almacenamiento. Las imágenes inferiores muestran las jerarquías de memoria en la arquitectura CUDA.

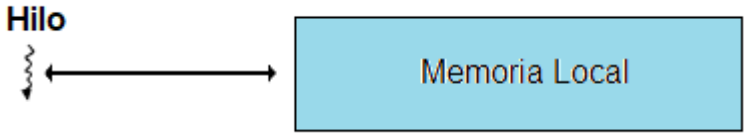


Figura 12: Memoria local

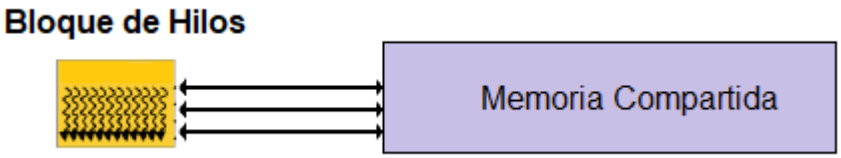


Figura 13: Memoria compartida

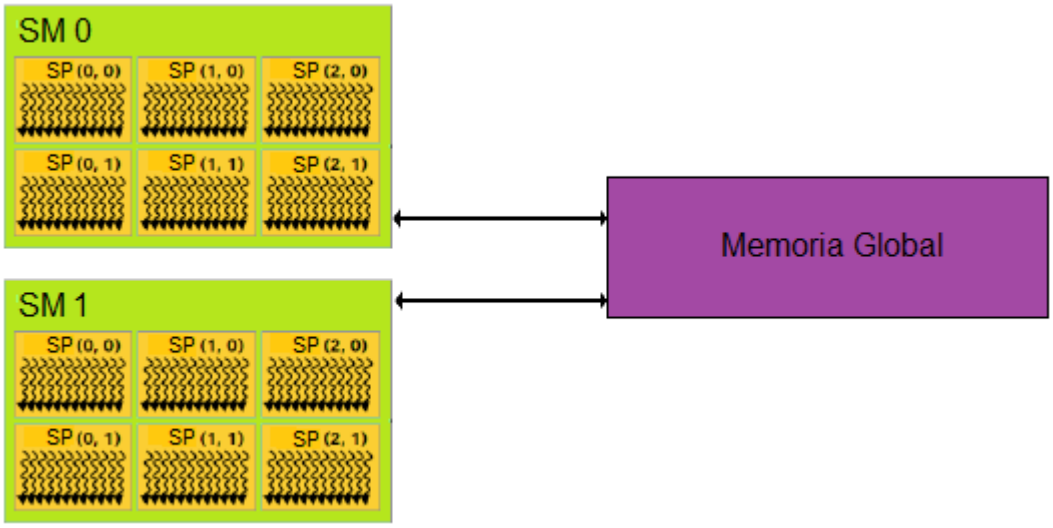


Figura 14: Memoria local

Los hilos están sincronizados y se organizan en bloques, y estos a la vez en mallas según su posición jerárquica. Estas mallas son ejecutadas independientemente por lo que funcionan en paralelo dentro de cada SM, al trabajar de este modo y tener diferentes niveles se ha de especificar tanto el número de hilo y de malla. Por lo que cada hilo de la GPU tiene asignado un número de identificación dentro de su malla, y cada malla está enumerada dentro de su SM.

En la imagen siguiente se muestra un esquema de lo explicado anteriormente.

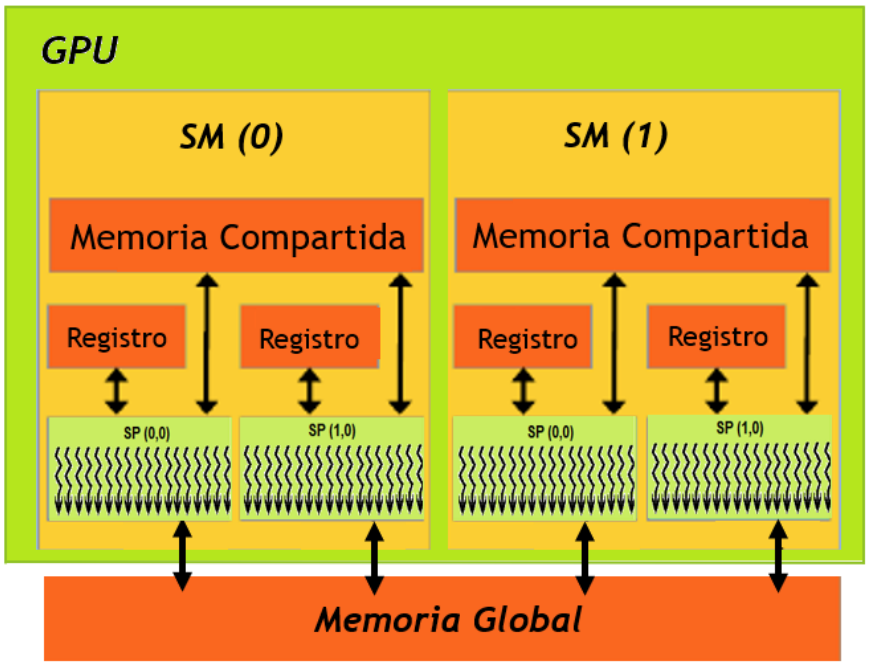


Figura 15: Jerarquía GPU

Cuando la GPU recibe una tarea la fragmenta en trozos, y cada uno de sus hilos la procesa de manera independiente. Esto se realiza mediante una unidad de distribución de trabajo, la cual selecciona un bloque SM, dentro de él a una malla que esté libre y envía las instrucciones a todos

sus hilos, mediante un doble mapeo de hilos y mallas. Cada núcleo CUDA se ejecuta de manera independiente y a su salida se unen todos los fragmentos obteniendo así la tarea completada.

La memoria compartida a diferencia de la memoria global tiene un elevado ancho de banda y baja latencia, así que, para acelerar el procesamiento de datos, se ha de copiar en la memoria de cada SM los datos utilizados. La GPU, aunque tiene 256 núcleos no se pueden utilizar todos ellos a la vez, ya que cada SM tiene un número determinado de registros por lo que sólo es posible activar un número máximo de núcleos.

La mayoría de las funciones que realiza la GPU se llevan a cabo mediante la GPC (*Graphics Processing Clusters)* [29], quien realiza el cálculo, rasterización, sombreado y texturizado de las imágenes. Por ello consta de las siguientes unidades:

- Las unidades de textura, la cual se encarga de realizar los filtros.
- Las unidades de carga y almacenamiento que captan y guardan los datos en la memoria.
- Los SFU (*Special Function Units*) que son unidades de funcionamiento especiales que manejan instrucciones de interpolación trascendental y gráficos.
- PolyMorph que se encarga de la obtención de vértices, teselado [59], transformaciones de ventanas gráficas, configuración de atributos y salidas de flujo.

CPU

Es la unidad central de procesamiento que se encarga de ejecutar un conjunto de instrucciones mediante operaciones lógicas y aritméticas. La CPU a diferencia de la GPU funciona en serie y posee 6 núcleos de procesamiento, divididos en dos clústeres:

- MCPU: Procesador Denver 2, de núcleo doble.
- BCPU: Procesador ARM [3] Cortex-A57, de núcleo cuádruple.

Ambos clústeres admiten ARMv8 de 64 bits (Aarch64) y de 32 bits (Aarch 32), y están conectados mediante un tejido de interconexión, permitiendo de este modo un funcionamiento simultáneo de ambos. Estos dos clústeres se conectan a MSelect FIFO [42] mediante una interfaz AXI (*Interfaz Extensible Avanzado*) [4], quien se encarga de organizar y seleccionar las entradas y salidas. Mediante este interfaz se envía las instrucciones a los buses periféricos según la dirección indicada. El puente de conexión entre AXI y Xbar [67] permite una respuesta rápida y un máximo rendimiento de transición a MMIO (*Mapeo de Memoria de I/O*) [41].

Procesador Denver 2 (núcleo doble)

El procesador Denver 2 con arquitectura ARM, permite un mayor rendimiento mono-hilo, obteniendo una mayor optimización dinámica. Se encarga de realizar dos funciones que son la decodificación hardware simple y la traducción binaria software. Estas operaciones se pueden realizar en otro momento en el que exista menos carga de trabajo, gracias a que cuenta con una memoria cache [39] de 128 KB de L1. Esta memoria se encarga de almacenar este conjunto de código y llevarlo a cabo posteriormente con el resto de los códigos pendientes.

Cada núcleo de este procesador Denver contiene:

- 128 KB de instrucciones (I-cache) [35].
- 64 KB de datos (D-cache) [14] de L1 [37].
- Ambos núcleos comparten 2MB de L2 de 16 vías.

A continuación, se muestran las características más destacables:

- Posee un búfer de bifurcación dinámica, una memoria RAM de almacenamiento global y un búfer de pila de retorno.
- Cuenta con 128 entradas de 4 segmentos asociados a instrucciones de L1 TLB (*Translation Lookaside Buffer*) [60] de 4 KB.
- Posee 256 entradas de 8 segmentos asociados a datos de L1 TLB de soporte 4 y 64 KB.
- Además, cuenta con 2048 entradas de 8 segmentos de sistemas asociados de aceleración TLB en cada procesador.
- 128 KB de 4 segmentos asociados a la protección de instrucción de paridad de L1.
- 64 KB de 4 segmentos asociados a la protección de datos de paridad de L1.
- PMU (*Performance Monitor Unit*).
- Interfaz con un controlador de interrupción genérico externo (Vgic-400).
- Sistema criptográfico [11] TSEC que se encarga de los algoritmos, protocolos y sistema de seguridad de la placa.

La parte hardware TSEC se encarga de comprobar y autentificar los códigos mediante Secure Boot ROM. El hardware del sistema se configura previamente, por lo que al iniciar el sistema el Boot Leader comprueba sí coinciden ambos códigos. Si el código es válido se pondrá en funcionamiento, por el contrario, si no coincide impide su puesta en marcha. Está compuesto por:

- Un generador de números aleatorios (RNG).
- Estándar AES-128b: Es un esquema de bloques que realiza cifrado y descifrado. Estos boques se operan em matrices de 4x4 bytes. Son rápidos, fáciles de implementar y no requieren de mucha memoria.
- Dos colas de instrucciones, que son utilizadas para el cifrado en sistemas DRM (*Digital Rights Management*) [20].

Procesador ARM Cortex-A57 (núcleo cuádruple)

Por otro lado, el procesador ARM Cortex-A57 se caracteriza por su uso en aplicaciones con múltiples subprocesos y cargas más ligeras. Este consiste en una microarquitectura que implementa el conjunto de instrucciones ARMv8-A de 64 bits. Está situado en núcleos SIP (*System in a Package*) [44] cuyo conjunto de varios de estos forman una matriz que constituye un SoC.

Al igual que el procesador Denver, este está basado en la arquitectura ARMv8, en los que cada núcleo contiene:

- 48 KB de instrucciones (I-cache).
- 32 KB de datos (D-cache) de L1.
- Todos los núcleos compartes 2MB de L2.

Entre sus características, las que más destacan son las siguientes:

- Tiene 48 entradas totalmente asociados a instrucciones de L1 TLB con soporte de 4 KB, 64 KB y 1 MB.
- Cuenta con 32 entradas totalmente asociados a datos de L1 TLB con soporte de 4 KB, 64 KB y 1 MB.
- 1024 entradas de 4 segmentos asociados al L2 TLB en cada procesador.
- 48 KB fijos de protección de paridad en las instrucciones de L1 y 32 KB de memoria ECC (*Error Correcting Code*) de datos de protección en el L1. Estas memorias ECC tienen la capacidad de detectar y corregir errores, recuperando de este modo el funcionamiento del sistema ante un fallo.

Tanto la frecuencia de la GPU y CPU se pueden modificar mediante la ejecución de unos determinados comandos (JetsonHacks, 2017) [40]. En la siguiente tabla se muestran los valores de sus frecuencias según el comando utilizado.

Modo	Nombre del Modo	CPU				Frecuencia GPU
		Denver 2	Frecuencia	ARM A57	Frecuencia	
0	Max-N	2	2.0 GHz	4	2.0 GHz	1.30 GHz
1	Max-Q	0	-	4	1.2 GHz	0.85 GHz
2	Max-P Core-All	2	1.4 GHz	4	1.4 GHz	1.12 GHz
3	Max-P ARM	0	-	4	2.0 GHz	1.12 GHz
4	Max-P Denver	2	2.0 GHz	0	-	1.12 GHz

Tabla 3: Frecuencia de la CPU y GPU

De este modo se consigue aumentar la frecuencia de reloj de muestreo de ambas y la activación o desactivación de sus respectivos núcleos.

Controlador de memoria (MC)

El uso de este controlador permite aumentar el rendimiento de la memoria y proporciona una mayor velocidad a las instrucciones de la CPU. Su funcionamiento se basa en la priorización de las solicitudes de instrucciones, en la optimización de la eficiencia, en el acceso a la memoria de todos los dispositivos internos y en minimizar el consumo de energía. La estructura del subsistema de memoria MSS (*Maximum Segment Size*) [43] está compuesto por 4 componentes:

- Columna MSS: Se encarga de las solicitudes y respuestas entre el MC y los usuarios.
- Centro MC: Realiza varias funciones ya que se encarga de recibir las solicitudes, realiza comprobaciones de seguridad y envía la respuesta a los cuatro canales MC.
- Canales MC: son las filas de clasificación y control de la DRAM (*Memoria Dinámica de Acceso Aleatorio*) [21].
- DRAMIO: es el canal de tela de protección DRAM de entradas y salidas, y PLLs (*Phase Locked Loop*) [47].

Algunas de sus características principales son las siguientes:

- Interfaz de memoria de 128 bits.

- Soporte para el cifrado de la DRAM, cumpliendo así los requisitos de seguridad SCSA (*Signal Computing System Architecture*) [49].
- Su direccionamiento virtual es de 40 bits.
- Señal dual CKE (Clock Enable) para un apagado dinámico por dispositivo.
- Soporte para 2 rangos de DRAM con densidades desiguales.

Esta tarjeta TX2 no requiere de relojes externos para su funcionamiento, debido a que todos sus relojes se encuentran dentro del módulo, incluyendo el reloj de baja potencia de 32.768 kHz. Posee un oscilador de 38.4 MHz el cual se utiliza como el reloj de referencia para la mayoría de los PLLs del sistema. Estos PLLs se utilizan para generar un reloj de alta frecuencia el cual se encarga de manejar datos paralelos de 10 bits por segmento a la velocidad indicada.

Módulo de memoria

Memoria LPDDR4 de 8 GB

Esta memoria DRAM de 128 bits es la generación siguiente a DDR4 y es utilizada para el almacenamiento de datos a corto plazo. Las principales diferencias entre estas dos versiones se muestran en la siguiente tabla.

	DDR4	LPDDR4
Velocidad de datos	1600 Mb/s – 3200 Mb/s	400 Mb/s – 3200 Mb/s
Densidad de chips	4 Gb – 16 Gb	8 Gb – 32 Gb
Número de banco	4	8
Frecuencia de transmisión	1.6 GHz-667 MHz	1.6 GHz
Voltaje	1.2 V	1.1 V

Tabla 4: Diferencia entre DDR4 y LPDDR4

El módulo TX2 cuenta con la LPDDR4 lo que le permite doblar la velocidad de procesamiento de datos y reducir el consumo de potencia. Alcanza una velocidad de 3732 MT/s, con un ancho de banda de 59,7 GB/s, implementado en cuatro segmentos de 32 bits de memoria, con subparticiones de 16 bits por canal. Esto hace que las rutas de transferencia sean más cortas, reduciendo de este modo la potencia del núcleo y aumentando la velocidad del sistema. Otra de sus mejoras es el modo de ahorro de energía que funciona al realizar tareas simples. Al llevar a cabo tareas que no requieren de un elevado consumo, la tensión disminuye de 1.2 V a 1.1 V. Al no ser necesario que el reloj funcione a alta velocidad, este reduce su valor permitiendo un ahorro en el consumo de la batería.

Módulo eMMC 5.1 (Embedded MultiMediaCard) [25]

Es una memoria no volátil de 32 GB que cuenta con un controlador de almacenamiento y se caracteriza por sus elevadas tasas de transferencia de datos, siendo la de lectura 250 MB/s y la de escritura 125 MB/s. Su interfaz es de 8 bits y se encarga del almacenamiento de datos, reduciendo de este modo la carga de trabajo de la CPU. (Jedec.org, 2015) [41] (Luis, 2015) [53]

4.1.2 Sistema de control de la tensión

El suministro de energía de este módulo es controlado por dos monitores de control compatibles con el I2C. Tienen dos señales WARN y CRIT que se activan al detectar cualquier condición que no se ajuste dentro del rango permitido de funcionamiento. Cada controlador tiene un circuito electrónico, que difiere del otro según el valor de sus resistencias, conexión de la señal AO y señales de los componentes conectados. Dependiendo de esto último se utilizará uno u otro, sus principales características son:

- La tensión mínima de alimentación es de 2.7 V, siendo la máxima 5.5 V.
- El rango de temperatura de funcionamiento es -40º a 125ºC.
- Compuesto por 16 pines.

En la siguiente tabla se muestra una descripción de sus pines. (INA3221 Triple-Channel, High-Side Measurement, Shunt and Bus Voltage Monitor with I 2C- and SMBUS-Compatible Interface, 2016) [38]

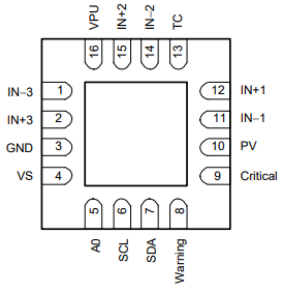


Figura 16: INA3221

Pin	Nombre		Dirección	Descripción
1	IN -3	VIN3N	Entrada Analógica	Unión entre la resistencia y tierra
2	IN +3	VIN3P	Entrada Analógica	Conexión con un lado de la resistencia para su suministro
3	GND	GND	Masa	Masa
4	VS	VS	Alimentación	Suministro de tensión
5	A0	A0	Entrada Digital	Byte de direccionamiento
6	SCL	SCL	Entrada Digital	Bus serie de reloj, entrada drenaje abierto
7	SDA	SDA	Entrada y Salida Digital	Bus serie de datos, entrada/salida drenaje abierto
8	Warning	WARN	Salida Digital	Señal de advertencia
9	Critical	CRIT	Salida Digital	Señal de alerta crítica
10	PV	PV	Salida Digital	Señal válida de potencia
11	IN -1	VIN1N	Entrada Analógica	Unión entre la resistencia y tierra
12	IN +1	VIN1P	Entrada Analógica	Conexión con un lado de la resistencia para su suministro
13	TC	TC	Salida Digital	Alerta de control de tiempo
14	IN -2	VIN2N	Entrada Analógica	Unión entre la resistencia y tierra
15	IN +2	VIN2P	Entrada Analógica	Conexión con un lado de la resistencia para su suministro
16	VPU	VPU	Entrada Analógica	Tensión pull-up para la polarización

Tabla 5: Pines del INA3221

El módulo TX2 se comunica con el controlador mediante un byte de direccionamiento, este está formado por 8 bits, de los cuales 7 se utilizan para la dirección y el octavo bit para indicar si se trata de una lectura o escritura. Esta señal es recibida por el pin A0 del controlador, cuyo nivel lógico se ha de configurar antes de empezar a utilizar el controlador. En la siguiente tabla se muestran los niveles lógicos del pin A0 según los posibles direccionamientos.

Señal	Nivel lógico A0
GND	10000000
VS	1000001
SDA	1000010
SCL	1000011

Tabla 6: Niveles lógicos del A0

Este circuito se utiliza en el caso de la GPU, SoC y WLAN, en él la señal A0 está direccionada a GND y presenta las siguientes uniones:

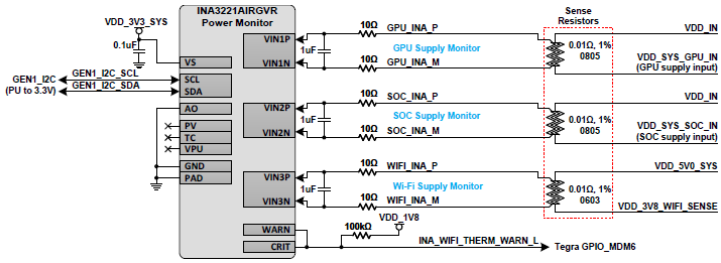


Figura 17: Control GPU, SOC y WLAN

Módulo controlado	Valor de resistencias	Pines de conexión
GPU	0.01 Ω	VIN1P Y VIN1N
SoC	0.01 Ω	VIN2P y VIN2N
WLAN	0.01 Ω	VIN3P y VIN3N

Tabla 7: Control GPU, SOC y WLAN

Este otro circuito se utiliza si se trata de la CPU, SRAM y VDD_IN. En este caso la señal A0 está direccionada con VS.

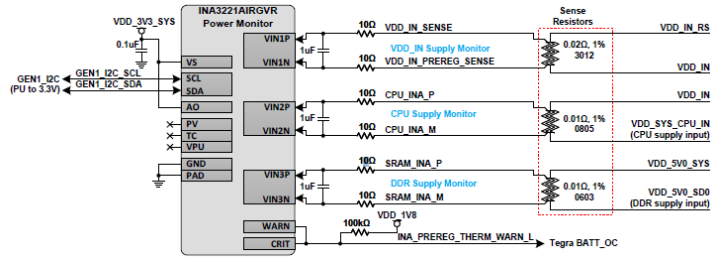


Figura 18: Control CPU, DDR y VDD_IN

Módulo controlado	Valor de resistencias	Pines de conexión
CPU	0.02 Ω	VIN1P Y VIN1N
SRAM	0.01 Ω	VIN2P y VIN2N
VDD_IN	0.01 Ω	VIN3P y VIN3N

Tabla 8: Control CPU, SRAM y VDD_IN

4.1.3 Estudio térmico

Para realizar este estudio térmico se ha tomado como referencia el documento (Thermal Design Guide, 2017) [92]. El sistema de refrigeración de la tarjeta Jetson TX2 está diseñado en base a su TMP (*Total Module Power*) [62]. El valor de la potencia del módulo varía según el componente y tarea que se esté realizando, por lo que las cargas y puntos de calor serán distintos. Con el sistema de refrigeración se consigue que la temperatura se propague y se distribuya de manera uniforme, obteniendo un mayor rendimiento térmico. Durante los ensayos de refrigeración se observó que la zona de la placa donde se alcanza la mayor temperatura es la señalada por (+) en la imagen inferior.

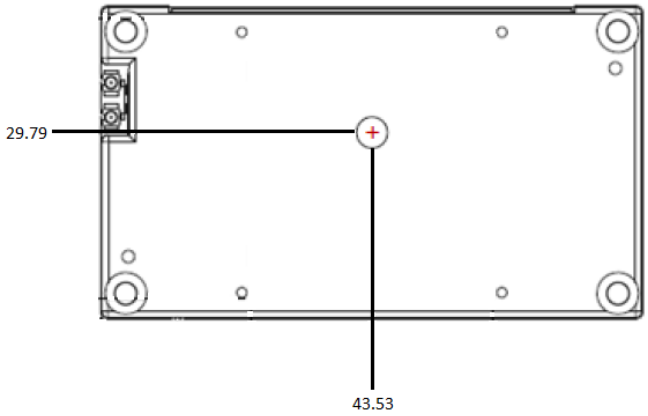


Figura 19: Punto de referencia

En la siguiente tabla se muestran sus especificaciones térmicas

Parámetro	Valor
Temperatura máxima de funcionamiento TTP	80 ° C
Límite de temperatura de funcionamiento recomendada	T.cpu = 95.5 ° C
	T.gpu = 93.5 ° C
Límite máximo de temperatura de funcionamiento	T.cpu = 101 ° C
	T.gpu = 101 ° C

Tabla 9: Temperaturas de funcionamiento

Como ya se ha dicho anteriormente con la tarjeta Jetson TX2 se realizaron una serie de pruebas y un reentrenamiento de la red neuronal. Durante uno de estos entrenamientos se realizó la siguiente captura de imagen que corresponde a las temperaturas que la GPU, CPU y otros componentes alcanzaban.

```
nvidia@tegra-ubuntu:~$ sensors
BCPU-therm-virtual-0
Adapter: Virtual device
temp1:        +58.0°C  (crit = +101.0°C)

MCPU-therm-virtual-0
Adapter: Virtual device
temp1:        +58.0°C  (crit = +101.0°C)

GPU-therm-virtual-0
Adapter: Virtual device
temp1:        +64.0°C

AO-therm-virtual-0
Adapter: Virtual device
temp1:        +62.0°C  (crit = -40.0°C)

Tboard_tegra-virtual-0
Adapter: Virtual device
temp1:        +53.0°C  (crit = +107.0°C)

Tdiode_tegra-virtual-0
Adapter: Virtual device
temp1:        +61.8°C  (crit = +107.0°C)

thermal-fan-est-virtual-0
Adapter: Virtual device
temp1:        +59.4°C
```

Figura 20: Temperatura durante el entrenamiento

Sistema de ventilación

Mediante el módulo TX2 se configuran los parámetros PWM (*Pulse Width Modulation*) y la señal tacométrica del ventilador. La temperatura referencia con la que el ventilador se activa es la del SoC, ya que la temperatura del resto de componentes es superior a esta. Además, cuenta con diferentes rangos de histéresis cuyo valor varía en función del rango de temperatura en el que se encuentra. La gráfica siguiente muestra la relación entre la temperatura de unión (Tj) y (PWM).

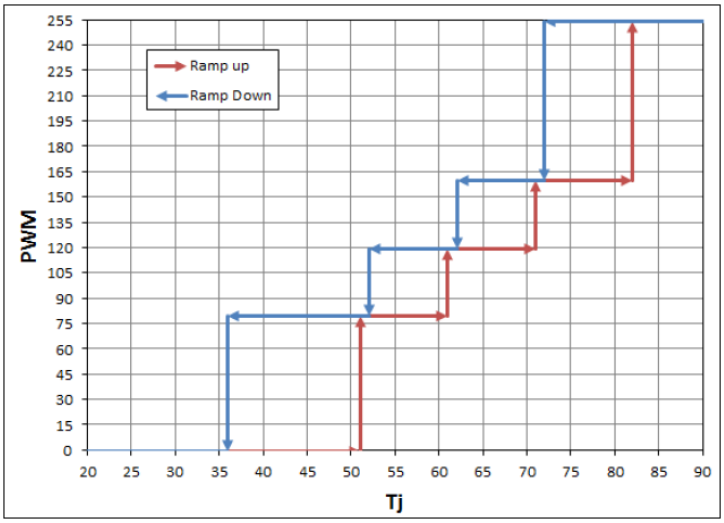


Figura 21: Gráfica Ti-PWM

La tabla inferior muestra algunos datos de temperatura del SoC con sus respectivos valores de histéresis. Siendo esta la temperatura que se toma como referencia, se observa que el ventilador empieza a funcionar cuando se alcanzan los 51 °C, y deja de funcionar cuando la temperatura es de 35 °C o inferiores.

Temperatura SOC (°C)	PWM	Histéresis (°C)
51	80	15
61	120	9
71	160	9
81	255	10

Tabla 10: Temperatura SoC

4.1.4 Placa de transferencia térmica (TTP)

Es una placa de 39 x 45 mm situada en la parte superior del módulo Jetson TX2, que mejora su rendimiento térmico, debido a la placa interna difusora de calor conectada a Tegra TX2. Se une al módulo mediante unos tornillos, permitiendo de este modo una mejor fijación. Se observa en la imagen que la placa TTP cubre por completo todos los componentes del módulo, dejando sin cubrir únicamente los conectores de las antenas Wifii.

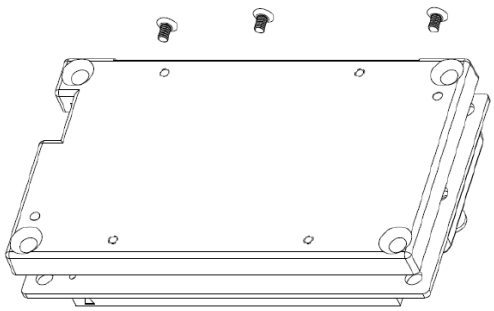


Figura 22: Placa TTP

Este bloque está compuesto por distintos componentes que son los siguientes:

- PCB: Circuito impreso del módulo TX2 al que están conectados sus componentes.
- TX2: Representa a los componentes del módulo TX2, en concreto la zona señalada con el 1 indica la CPU y GPU.
- TIM: Son materiales de interfaz térmico que se utiliza para aumentar el rendimiento de la dispersión y transferencia de calor. El utilizado en este módulo es el GF3500S35 que es un gel de baja viscosidad, que rellena los espacios vacíos o con aire entre la unión de los dos componentes.
- TTP: Es una placa de transferencia térmica colocada en la parte superior e inferior, como se observa en la imagen. Tiene internamente una placa difusora de calor conectada a Tegra TX2, para conseguir un rendimiento térmico estable.
- Termopar: Colocado en la superficie de la placa TTP, indicado por el número 2. Este transductor está formado por dos metales distintos que en función de la diferencia de temperatura generan una diferencia potencial.

- HS_TIM: Material de interfaz térmico utilizado para la unión entre la placa TTP y el disipador de calor.
- Thermal Solution: Representa al componente capaz de enfriar y controlar el TMP del sistema. En este caso se ha utilizado un disipador de calor.

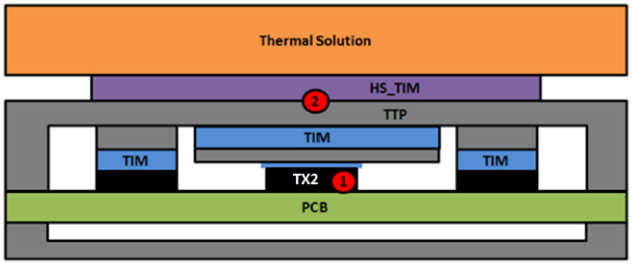


Figura 23: Componentes térmicos

En la parte superior de la placa TTP se encuentra un disipador de calor de aluminio plateado que cubre toda su superficie. El utilizado en esta placa es el modelo DVC-01672-N2-Gp, el cual lleva en su centro un pequeño ventilador negro. (DCV−01672−N2−GP,2018) [10]

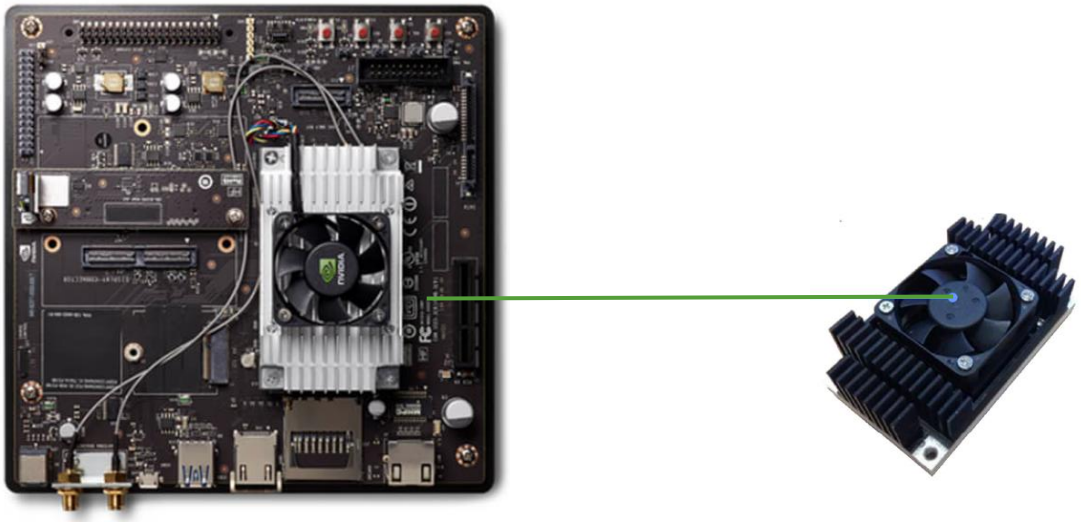


Figura 24: Disipador de calor

El suministro de tensión al ventilador se realiza mediante un conector con 4 pines, en la siguiente tabla se muestra la descripción de estas señales.

Número Pin	Señal	Descripción	Direccionamiento
1	GND	Conexión a masa	Tierra
2	VDD_5V_IO	Alimentación	Alimentación
3	FAN_TACH	Señal del tacómetro	Entrada
4	FAN_PWM	Señal de modulación de ancho de pulso	Salida

Tabla 11: Señales del disipador de calor

La siguiente imagen muestra la conexión interna de este conector.

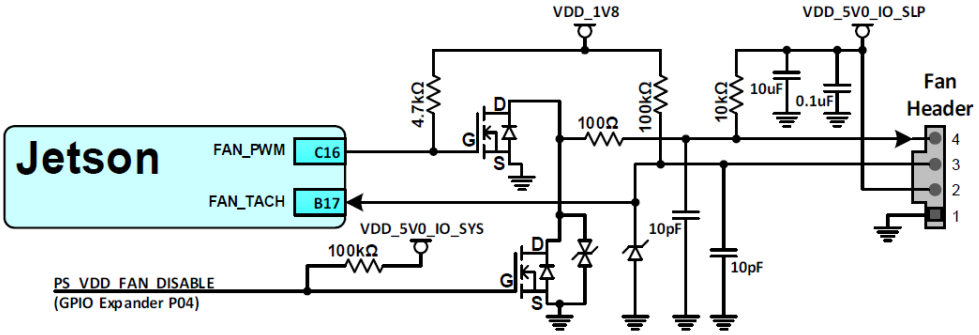


Figura 25: Conexión del disipador de calor

Sensores de temperatura

Esta placa cuenta con múltiples sensores de temperatura colocados en los lugares donde se concentra mayor cantidad de calor. Se encarga de controlar la temperatura de unión TIM y TTP, y de otras funciones como la del termopar o activar mecanismos de protección. La lectura de estos sensores y la activación del modo de protección se puede realizar mediante software y hardware.

Modo Software-Thermal Throttling

El modo de funcionamiento del controlador software depende del valor de la temperatura que detecten los sensores. Existen dos casos:

- Si la temperatura es igual o inferior a 80 °C: en este caso el controlador esta desactivado y el sistema funciona libremente, sin que este intervenga.
- Si la temperatura supera el umbral programado: los sensores térmicos señalan que se ha superado el límite de la temperatura y se genera una interrupción. Por lo que el algoritmo de control térmico se activa y realiza periódicamente las siguientes operaciones:
  - Temperatura de sondeo.
  - Ejecuta el algoritmo de control y calcula la disminución de la velocidad de reloj que se ha de aplicar en el próximo periodo de tiempo.
  - Regula el sistema para disminuir la velocidad, y se repite los pasos anteriores hasta que el sistema alcance una temperatura inferior a la del umbral.

Modo Hardware -Thermal Throttling

Este modo conlleva la reducción de un mayor rendimiento del sistema y al igual que el modo anterior cuenta con un mecanismo que disminuye la velocidad de reloj, que en este caso es hardware. Este se encarga de desacelerar los relojes de la CPU y GPU para reducir de este modo la carga y evitar el apagado por sobrecalentamiento. Esto es el último recurso, ya que la máxima prioridad es evitar que el sistema se sobrecaliente y se apague.

Modo de protección

En ningún momento se ha de exceder la temperatura límite de 105 °C, ya que esto supondría la posible rotura del componente o que la placa deje de funcionar. Para evitar esto se produce el apagado inmediato del sistema, que se realiza mediante dos mecanismos. Estos no se pueden modificar, ya que son implementadas por NVIDIA garantizando así su correcto funcionamiento.

- Sensor interno: En él se utiliza una señal de apagado proveniente directamente desde PMIC. Después para establecer nuevamente su funcionamiento el usuario ha de encender manualmente el sistema mediante el botón de encendido.
- Monitor de temperatura: cuando este monitor detecta que la temperatura del diodo está por encima de la temperatura programada previamente, la salida THERM del monitor envía una señal al PMIC para que apague el sistema directamente, sin la necesidad del control software. Esta temperatura programada es superior a la temperatura umbral, evitando de este modo llegar a una situación límite.

4.1.5 Fuente de alimentación (DC Power)

La tarjeta es alimentada mediante el conector que está situado en uno de sus laterales, cuyo rango de voltaje es de 5.5 V a 19 V. Este voltaje es controlado mediante PMICs (*Power Management Integrated Circuits*), que controlan la energía a través de reguladores de voltaje e interactúan con el PMC (*Power Management Controller*) [46] mediante señales de banda lateral. En el caso de que la tarjeta no esté conectada a la fuente de alimentación, se le puede conectar una batería para mantener de este modo el sistema en RTC (*Real Time Clock*). Los valores de voltaje más comunes de estas baterías son 2.5 V y 3.5 V, pero se ha de tener en cuenta que el voltaje suministrado influye en la eficiencia energética.

Tiene dos modos de funcionamiento, uno para aquellas tareas que requieran alta velocidad y otro de baja potencia para los estados de espera. Su funcionamiento principal es controlar las transiciones de voltaje del procesador Tegra, ya que se transfiere desde diferentes modos de baja potencia. El PMC recibe señales de requerimiento de tanto energía, reloj y de eventos, este último para llevar a cabo la activación de fuentes como el I2C y RTC, los cuales pueden activar al procesador Tegra.

El PMC se basa en una determinada lógica que mantiene definidos los estados y controla los dominios de potencia durante los estados inactivos. Cuando se activa la señal RESET del PMC, este genera reinicio para PMC, RTC y CAR (*Clock and Reset*) [7], al producirse el reseteo de este último se genera el reseteo de la mayoría de los bloques del chip. Por otro lado, el módulo RTC se mantiene siempre activo, lo que le permite habilitar el temporizador y el funcionamiento de los dispositivos externos cuando el sistema está en baja potencia.

Rieles de Potencia

La tarjeta Jetson TX2 posee sólo un conector de alimentación, mediante el cual se suministra la energía necesaria para el resto de las interfaces del sistema. Se ha de tener en cuenta que cuanto menor diferencia de voltaje exista, mayor puede llegar a ser su eficiencia. Tiene dos monitores

de potencia que se encargan de medir el consumo de energía de seis rieles. Uno de los monitores mide los rieles VDD_IN, VDD_DDR y VDD_WIFI, mientras que el segundo mide los rieles de VDD_SYS_CPU, VDD_SYS_GPU y VDD_SYS_SOC.

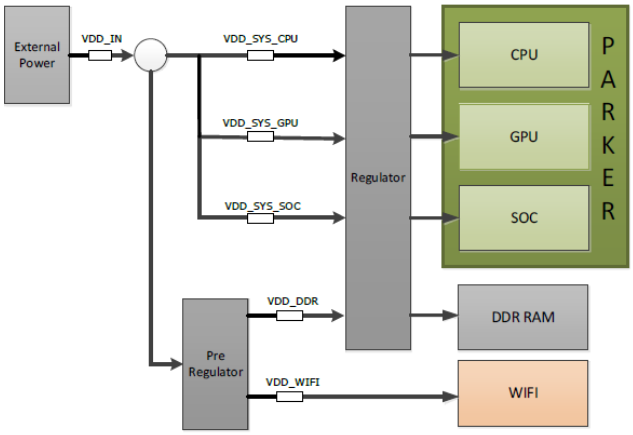


Figura 26: Conexión de los rieles

Todos los niveles de voltaje excepto VDD_WIFI y VDD_DDR, dependen del voltaje de la fuente de alimentación externa, siendo estas dos únicas dependientes del preregulador. Se ha de tener en cuenta que hay una pérdida de potencia debido al regulador que se encuentra previamente a la CPU, GPU, SoC y DDR RAM. Por lo que las mediciones han de incluir estas pérdidas del regulador, mientras que los otros rieles muestran los valores de potencia real. El funcionamiento de estos monitores consiste en determinar los valores de voltaje y corriente de estos seis rieles, por lo que realiza la media de las últimas 512 muestras de los datos obtenidos, y calcula la potencia. La imagen inferior muestra la conexión con los botones.

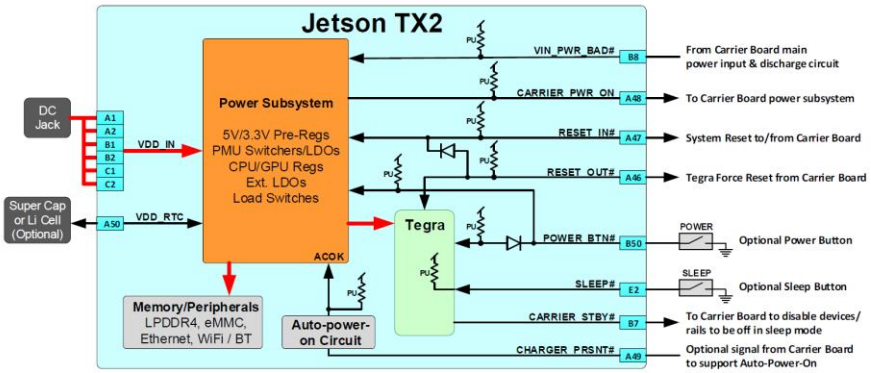


Figura 27: Conexión de los botones

Secuencia de potencia

Para evitar daños de los componentes debido a la potencia utilizada se ha de seguir una correcta secuencia de alimentación. Por ello la tarjeta Jetson TX2 genera una señal CARRIER_POWER_ON, que es transferida a Carrier Board, la cual indica que esta encendida y que el resto de los circuitos del Carrier Board pueden iniciarse.

Encendido

Durante el encendido, antes de habilitar al resto de componentes se debe esperar a que la señal CARRIER_POWER_ON se active. Cuando esta señal es confirmada se desactiva la señal RESET_OUT, habilitando al resto de componentes y permitiendo de este modo el arranque completo del sistema. La siguiente imagen muestra la secuencia de encendido del sistema.

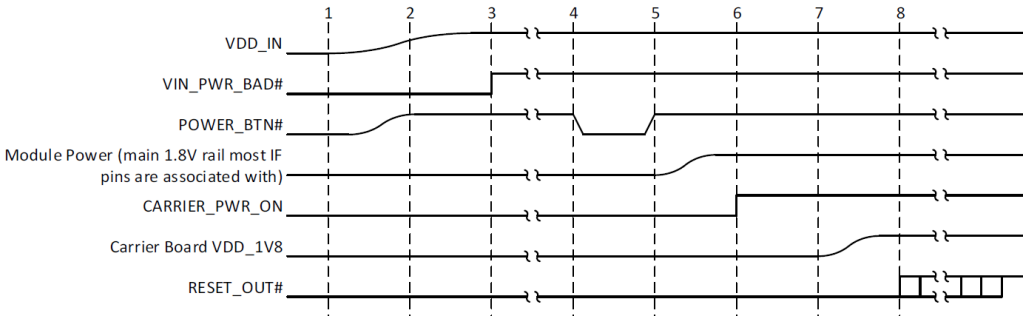


Figura 28: Secuencia de encendido

Apagado

Al recibir la señal Shutdown la placa activa la señal RESET_OUT, por lo que desactiva la señal CARRIER_POWER_ON. Esto hace que la tarjeta se apague y deshabilite al resto de componentes. Este apagado se realiza mediante la descarga de los distintos módulos, en la imagen siguiente se muestra la secuencia de apagado.

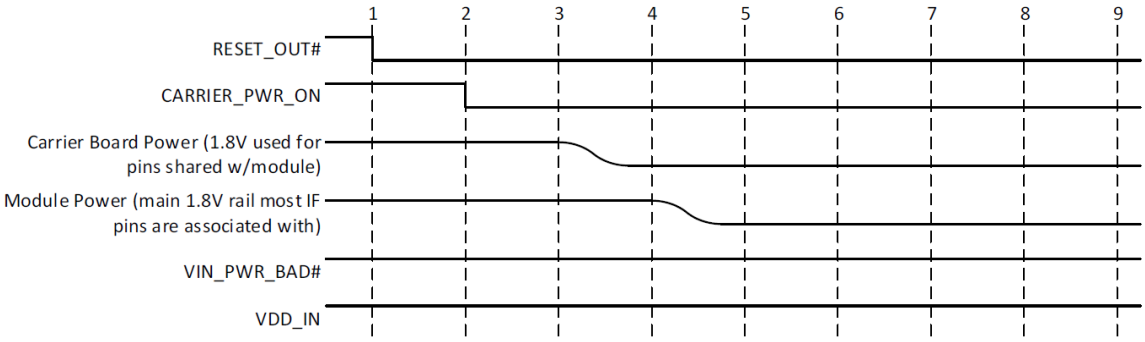


Figura 29: Secuencia de apagado

Modos de potencia

La tarjeta Jetson TX2 funciona en tres modos de potencia: OFF, ON y SLEEP.

Estado ON

Corresponde al funcionamiento normal de la tarjeta y se obtiene cuando la potencia de entrada es estable. Cuando se le suministra corriente a la placa, lo primero que hace es poner a uno la señal lógica POWER_BTN_BAD. Este uno lógico se mantiene mientras que la señal no sea estable. En el momento en el que se estabiliza se pone a cero, se desactiva y se enciende la tarjeta.

Estado OFF

En este estado todos los componentes están deshabilitados y la tarjeta apagada.

Estado SLEEP

Este estado corresponde cuando la tarjeta funciona a baja potencia. En él los componentes están lo suficientemente alimentados para permitir que el dispositivo se reanude y vuelva a entrar en el estado ON. Este estado se activa cuando no se produce ninguna activación durante un tiempo determinado. Para salir de este estado se debe producir un evento WAKE, el cual se puede producir desde el interior de la tarjeta o desde dispositivos externos.

4.1.6 Antenas

La tarjeta TX2 posee dos conectores de antena de doble banda, que se encuentran conectados al módulo WLAN/BT. Este módulo permite una conexión inalámbrica y combina dos estándares de datos como son Wifi y Bluetooth. Estos comparten la misma banda de frecuencia y su módulo combinado incorpora la conmutación periódica de la antena, permitiendo que ambos operen al mismo tiempo de un modo efectivo. Más adelante será explicado con mayor detalle.

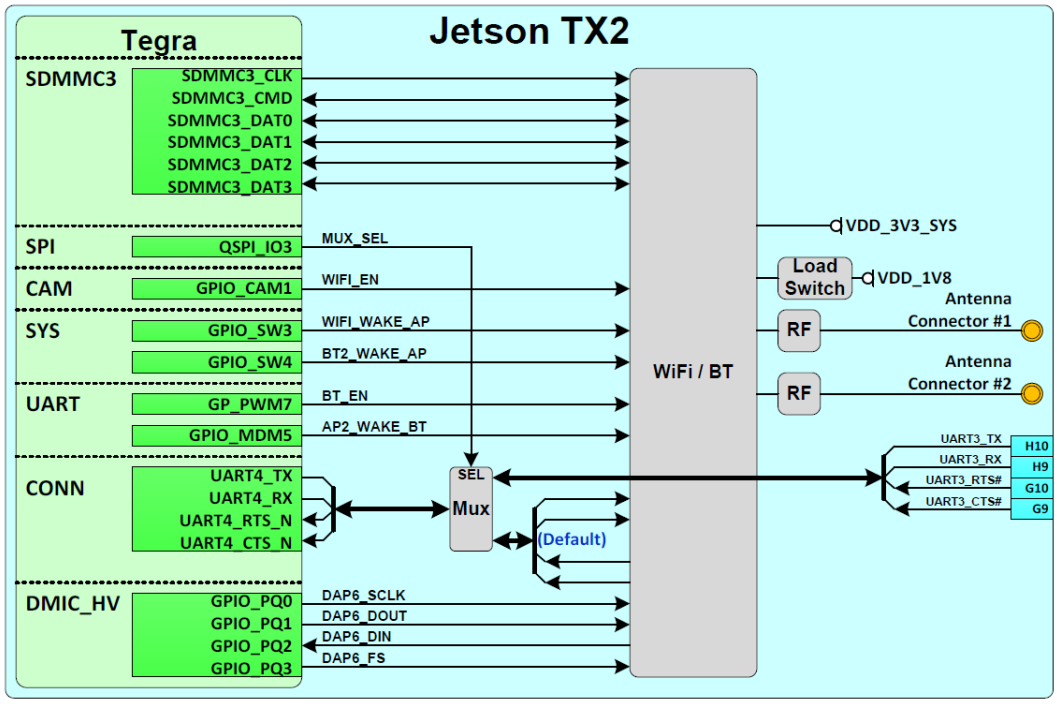


Figura 30: Conexión interna de las antenas

A continuación, se muestra una tabla con las características de su funcionamiento.

Parámetro	Dato
Tipo de antenas	Banda doble
Frecuencia de banda	2.4 y 5 GHz
Impedancia	50 Ω
Conector de acoplamiento	I-PEX U. FL

Tabla 12: Características de las antenas

4.1.7 Conexión USB micro 2.0

La tarjeta tiene un puerto de tipo USB 2.0 siendo su velocidad máxima de transferencia de aproximadamente 480 MB/s. Si se trata de una gran cantidad de datos puede llegar a existir un atasco en la transmisión de estos.

Admite tres modos de funcionamiento:

- Modo dispositivo: Únicamente se comunica con el controlador host cuando este lo solicita.
- Modo host: Permite comunicarse con todos los dispositivos USB, ya que realiza una transferencia de datos descendente. Esto consiste en reenviar los datos recibidos a los puertos descendentes, mediante una multiplexación y reenvió de información.
- Recuperación USB: Permite restaurar la unidad USB a su estado original e instalar el sistema operativo con la configuración y datos principales.

En general este puerto funciona a dos velocidades, una es cuando se le conecta un periférico USB 2.0, en el que su velocidad es de 480 Mb/s. Su segundo modo de funcionamiento es a 12 Mb/s, que corresponde a la conexión de un periférico USB 1.1.

Las características principales son:

- Velocidad máxima de transferencia de 480 MB/s.
- Modo de transmisión Half Duplex.
- Tiene 4 señales de conexión.
- La longitud máxima del cable es de 5 metros.
- El valor de la intensidad varía entre 100 mA y 500 mA.

El cable utilizado para la transmisión de datos (D+ y D-) es un par trenzado, en el que los dos hilos se utilizan tanto para la transmisión y recepción, pero de manera no simultánea. Utilizar este modo de transmisión diferencial reduce el efecto del ruido electromagnético interno de los hilos y permite mayores longitudes de cable.

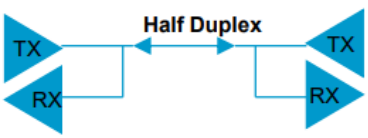


Figura 31: Half-Dúplex

4.1.8 Conexión tipo A USB 3.0

La tarjeta incorpora entre sus módulos un puerto de este tipo cuya velocidad alcanza los 5 GB/s, es decir es 10 veces más rápido que el USB 2.0. Sin embargo, a diferencia del anterior este sólo admite el modo host. Además, ofrece un mayor aporte energético ya que si un dispositivo conectado a él no se está utilizando, al cabo de un cierto tiempo se pone automáticamente en bajo consumo.

Las características principales son:

- Su velocidad máxima de transferencia es de 5 GB/s.
- Modo de transmisión Full Dúplex.
- Tiene 8 señales de conexión.
- La longitud máxima del cable se reduce a 3 metros.
- El valor de la intensidad varía entre 150 mA y 900 mA.
- Señales diferenciales adicionales RX y TX.

Al añadir un par de hilos adicionales permite que dos líneas envíen (RX) y otras dos reciban (TX), haciendo posible una transferencia de datos bidireccional simultánea.

Figura 32: Full-Dúplex

En la imagen inferior se muestra la conexión interna de los puertos USB 2.0 y 3.0.

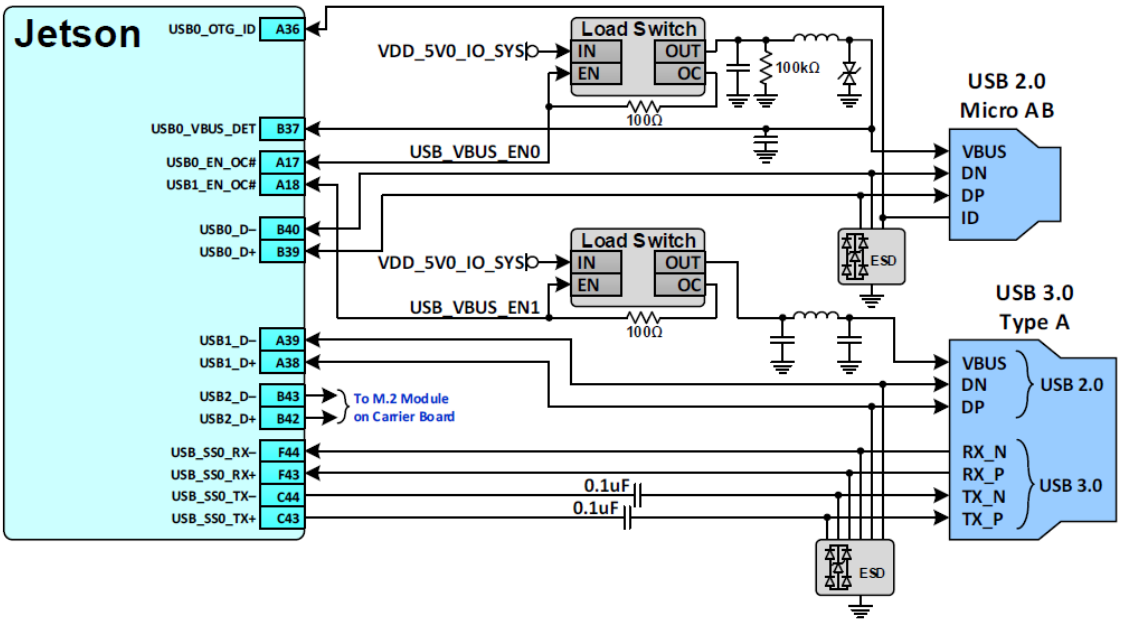


Figura 33: Conexión USB 2.0 y 3.0

En la tabla inferior se muestra las señales de los puertos USB 2.0 y 3.0.

Señal	Pin del módulo	Descripción	Tipo de direccionamiento
USB 2.0 Micro AB			
VBUS	USB0_VBUS_DET	Bus de alimentación	Alimentación
USB_IO_CONN_D_N	USB_D-	USB 2.0 Datos -	Bidireccional
USB_IO_CONN_D_P	USB_D+	USB 2.0 Datos +	Bidireccional
GND	-	Señal de Tierra	Masa
USB 3.0 Tipo A			
VBUS	USB0_VBUS_DET	Bus de alimentación	Alimentación
USB1_D_N	USB1_D-	USB 2.0 Datos -	Bidireccional
USB1_D_P	USB1_D+	USB 2.0 Datos +	Bidireccional
GND	-	Señal de Tierra	Masa
USB3_RX1_N	USB_SS0_RX-	USB 3.0 Recibe -	Entrada
USB3_RX1_P	USB_SS0_RX+	USB 3.0 Recibe +	Entrada
USB3_TX1_N	USB_SS0_TX-	USB 3.0 Transmite -	Salida
USB3_TX1_P	USB_SS0_TX+	USB 3.0 Transmite +	Salida

Tabla 13: Señales de los USB 2.0 y 3.0

4.1.9 Módulo M.2 de conectividad E

Este módulo posibilita que el dispositivo sea capaz de integrar múltiples funciones y se utiliza como interfaz para conexiones WLAN/BT, PCIe, USB 2.0, UART, I2S e I2C.Esta formado por 75 pines distribuidos a ambos lados, con 8 tomas de tierra.

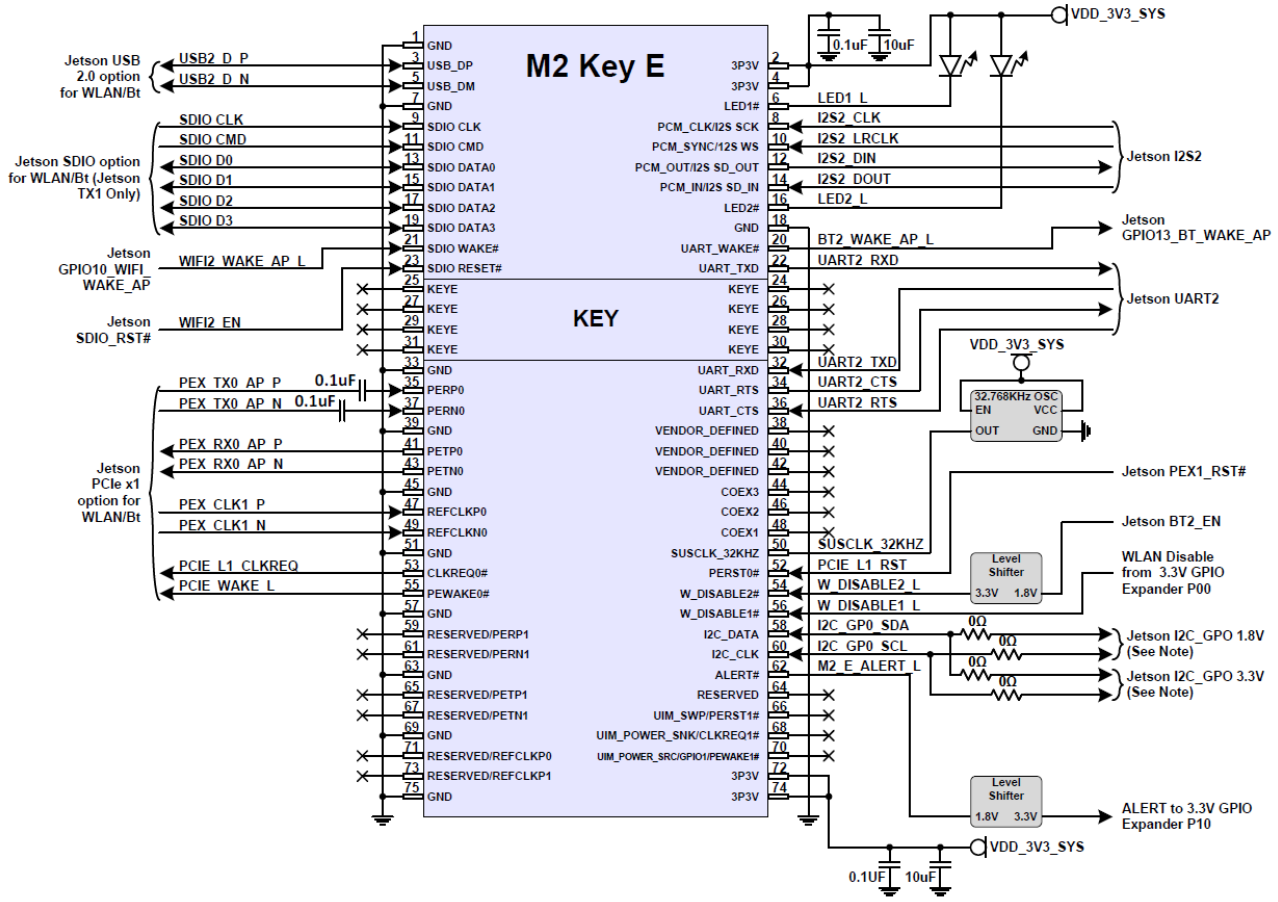


Figura 34: Conexión interna M.2 E

WLAN

El sistema WLAN es inalámbrico y transfiere la información mediante ondas de radio, por lo que no es necesario un medio físico. La transmisión se realiza mediante modulación de portadora, en el que las ondas transmitidas se envían a diferentes frecuencias. Esto permite que puedan existir diferentes ondas de transmisión al mismo tiempo, por lo que al receptor se le ha de indicar la frecuencia de la onda que ha de detectar. De este modo sólo escucha la onda que le interesa e ignora al resto.

Su estándar es el IEEE 802.11 ac con una banda de 5 GHz y una velocidad de transferencia de datos de 433 Mb/s. También es compatible con los estándares 802.11 b, g, a y n cuya banda de frecuencia es de 2.4 y 2.5 GHz. Otra de sus características es la multiplexación espacial 2x2 MIMO, esto indica que posee dos antenas de transmisión y recepción. Su funcionamiento consiste en multiplexar las señales para conseguir transportar más cantidad de datos, por lo que

cuando la información llega a la antena receptora, esta decodifica la señal y obtiene los datos enviados por la antena emisora. A diferencia de un sistema cableado la velocidad de transmisión es inferior, ya que al tratarse de un sistema inalámbrico se ha de realizar el cifrado de los datos a enviar y su descifrado al recibirlos. Este sistema es compatible con el modo STA (*Station*) y con los estados de potencia P2P (Peer to Peer).

Modo Estación STA (Station)

Posibilita la unión entre un dispositivo Wifi y uno de modo AP (*Access Point*) o WAP (*Wireless Access Point*) [66]. Se encarga de establecer los parámetros de funcionamiento de la red, centralizar y gestionar todas las comunicaciones inalámbricas. Existen tres modos de funcionamiento:

- PM0: Siempre está encendido.
- PM1: El equipo estación configura el bit PM e indica al punto de acceso que está entrando al modo de ahorro de energía, por lo que se activa periódicamente DTIM (*Delivery Traffic Indication Message*) [23] para verificar la señal AID. Si esta se establece, el STA utilizará un paquete PS-Poll para alcanzar los paquetes del búfer, resultando un método ineficiente.
- PM2: Se produce cuando existe una comunicación (Tx o Rx), en el que la señal DUT se activa y permanece activa durante todo el proceso de intercambio de datos. Cuando existe intercambio de datos este modo funciona como el modo PM0, mientras que si no hay intercambio de datos su funcionamiento es similar al del PM1.

Estado de potencia P2P (Peer to peer)

Este modo no realiza distinción entre host y cliente por lo que considera ambos nodos iguales. Estos se comportan a la vez como clientes y servidores, por lo que todos pueden realizar el mismo tipo de operaciones. Estos nodos son configurables independientemente ya que se les puede configurar la velocidad, el ancho de banda de conexión y la capacidad de almacenamiento.

El controlador de estado de potencia determina la categoría de acceso y las restricciones de una comunicación recibida de un componente P2P. Esta restricción determina el tipo de mecanismo de ahorro de energía del dispositivo.

- OPS (*Opportunistic Power Save*): Permite un ahorro de energía, aunque el resultado obtenido es bajo. Ya que este mecanismo solo se activa cuando los componentes están desactivados, siendo su implementación de baja complejidad.

- NoA (*Notice of Absence*): señala periodos de ausencia en los que las aplicaciones P2P no pueden acceder al canal, independientemente en el modo en el que estén. Por lo que la aplicación puede decidir de manera autónoma ponerse en modo de ahorro de energía. Para ello se ha de tener en cuenta los siguientes 4 parámetros:

  - Duración que determina el tiempo de cada periodo de ausencia.
  - Intervalo que especifica el tiempo entre los periodos consecutivos de ausencia.
  - Momento que especifica el tiempo de inicio del primer período de ausencia.
  - Contador que especifica cuántos períodos de ausencia se programan.

Bluetooth

Es un protocolo de comunicación que posibilita la transferencia de datos entre diferentes dispositivos inalámbricos mediante un enlace de radiofrecuencia en la banda ISM [5] de 2.4 GHz. La versión de esta tarjeta es la 4.1 y presenta las siguientes características:

- La reconexión automática de los dispositivos, posibilitando que un periférico se reconecte tras haber salido.
- Transmisión de lotes de datos más rápidos y eficaces.
- Conexión simultánea de dispositivos, los cuales se pueden utilizar como emisor y receptor al mismo tiempo.
- Mayor conectividad al internet de las cosas.

Otra de sus características es el compilador HIDP, que convierte el lenguaje paralelo de alto nivel en código fuente C++ de CUDA. Su objetivo es mejorar la codificación mediante el paralelismo de datos jerárquicos, para ello realiza un análisis interno en base a fragmentos o patrones de código para la computación en paralelo de datos de estructura jerárquica. Esto hace que no sea necesario la implementación de largos códigos y facilita recursos para la compatibilidad con arquitecturas paralelas modernas. El lenguaje utilizado en HIDP es independiente del sistema y en el caso de que exista diferentes opciones de mapeo el compilador genera para cada mapeo una versión de código diferente.

El controlador integrado Bluetooth contiene los siguientes estados:

- Activo: No se produce ahorro de energía, es el estado predeterminado.
- LP: Estado de ahorro de energía. En él, el chip BT pasa al modo de baja potencia y se activa cuando se produce un intercambio de datos. En el caso de que se reciba datos, el chip señala el accionamiento GPIO (*General Purpose I/O*) [30].

I2C

El interfaz I2C admite la comunicación serie con múltiples dispositivos, realiza el control del manejo del reloj y el control de las velocidades de los dispositivos normales y rápidos. Además, admite el direccionamiento esclavo de 7 bits similar que el protocolo I2C, al igual que el modelo de operación maestro-esclavo. Este posee dos señales SDA y SCL.

- SDA: Es la línea de datos serie semibidireccional. Es controlada por el emisor, el cual puede ser maestro o esclavo.
- SCL: Es la señal de sincronismo, en un esclavo esta señal es una entrada, mientras que en un maestro es una salida, el cual además evalúa su estado. Esta señal es gobernada únicamente por el maestro.

4.1.10 HDMI (High Definition Multimedia Interface)

Este conector es de tipo A por lo que cuenta con 19 pines y su frecuencia de 600 MHz por píxel. Su ancho de banda es de 18 Gb/s por lo que aumenta la resolución, la tasa de actualización y tiene mayor profundidad de color. Es compatible con las versiones 2.0 HDMI y DP [18] 1.2a. Se basa en el sistema FreeSync lo que adapta la frecuencia de actualización de la pantalla, evitando de este modo el colapso o desgarro provocado por la desalineación con la velocidad de fotogramas. Con esto se elimina el tiempo de espera que la interfaz del video requería para acabar el fotograma actual y el desgarro de pantalla al iniciar un nuevo fotograma. El formato de video que se puede transmitir es de 4K, es decir 3840 x 2160 pixeles a 60 Hz. Permitiendo una transferencia de color de 8 y 12 bits RGB y YUV444 [70], mientras que si se trata de YUV422 es de 8, 10 y 12 bits.

Posee el estándar Dual-Mode DisplayPort (DP++) lo que le permite utilizar conectores pasivos para conectarse a pantallas HDMI o DV. El dispositivo al detectar la conexión de un adaptador HDMI o DV transmite señales TMDS (*Transition Minimized Diferential Signaling*) [61] al pin 20 DP quien se encarga de proporcionar 3,3 V de potencia de corriente continua. Por lo que el adaptador de voltaje se encarga de cambiar el nivel de tensión de 3,3 V del DP a 5 V de la pantalla.

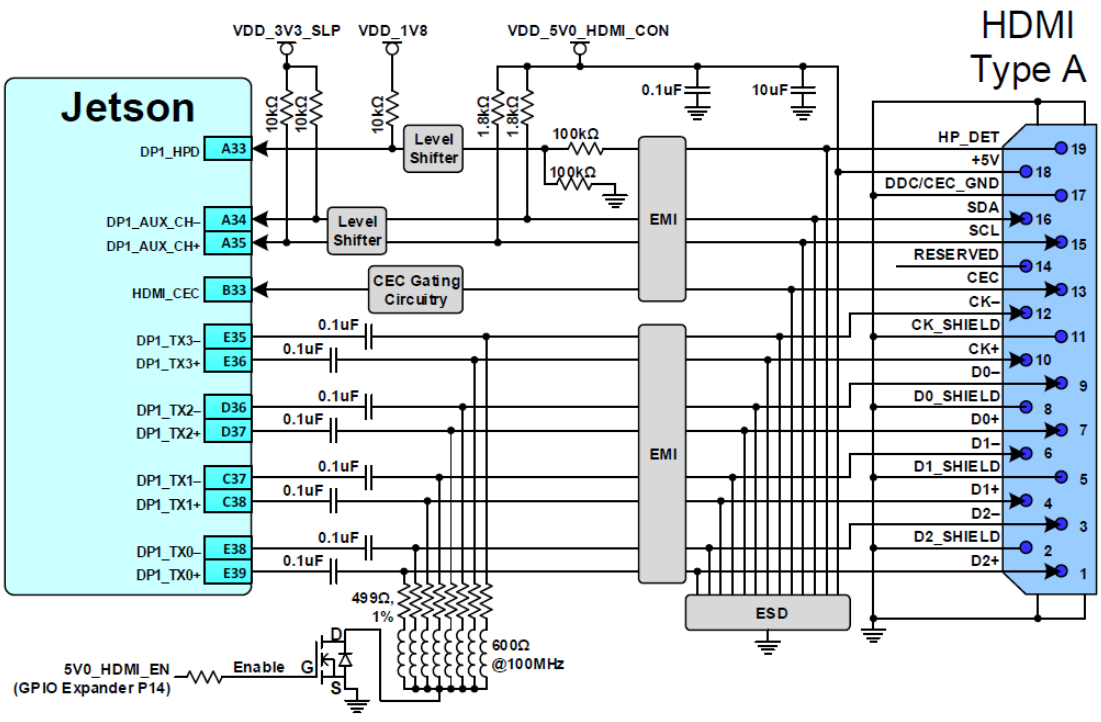


Figura 35: Conexión interna HDMI

4.1.11 Tarjeta SD

Este conector presenta el tamaño de 32 mm y es compatible con el modo SDR104 (UHS-1). Además, tiene un interfaz esclavo para acceder a la configuración de registros y a la RAM, ya que utiliza la ruta del sistema del controlador de memoria.

Al ser compatible con el modo SDR104 le permite llegar a una frecuencia máxima de 208 MHz, con una señal de 1.8 V y con una velocidad de bus de 104 MB/s. UHS-1 indica que tiene una señal de dato bidireccional, en el que interviene 4 pines de la tarjeta (7,8,9,1). La tarjeta es alimenta con 3,3 V y además es compatible con el modo SPI [57].

Algunas de sus características son:

- Interfaz de datos de 8 bits para el módulo eMMC y 4 bits para el módulo WLAN.
- Interfaz de datos de 4 bits para la tarjeta SD.
- Es compatible con tarjetas SD 3.0 o SDXC. La capacidad de las tarjetas SDXC es de 32 GB, cuya máxima teórica llega hasta los 2 TB.

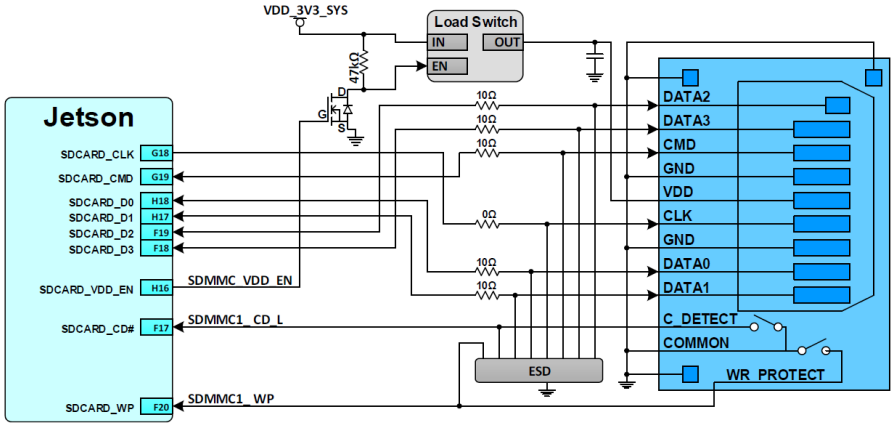


Figura 36: Conexión interna SD

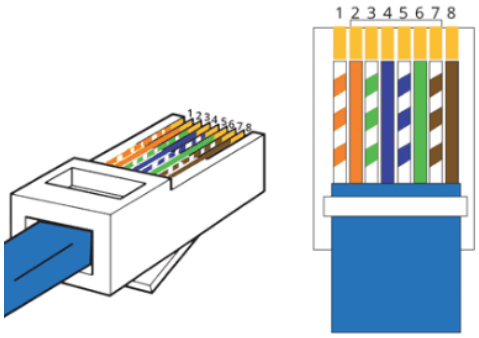
La siguiente tabla muestran algunas de las características de la tarjeta SD.

Características	SD
Socket SD	Sí
Pines	9
Anchura	24 mm
Longitud	32 mm
Grosor	2,1 mm
Compatible con el modo SPI	Si
Modo 1 bit	Sí
Modo 4 bits	Opcional
Modo 8 bits	No
Frecuencia del reloj	0 a 25 MHz
Máxima frecuencia	100 Mbit/s
Máxima frecuencia SPI	25 Mbit/s
DRM	Sí

Tabla 14: Características SD

4.1.12 Conector a Ethernet

Mediante este cable se conecta la tarjeta a internet, siendo esta conexión mucho más segura que la conexión Wifi a la hora de transmitir los datos y además se reducen las interferencias durante la transmisión. El conector de este cable es el RJ-45 y su interfaz cuenta con 8 pines, su modo de transferencia es por detección de la onda portadora y colisiones. Su estándar es el 802.3U de Base-T 10/100/1000 cuya capacidad es de 10/100/1000 Mb/s respectivamente. El tipo de cable utilizado es UTP (*Unshielded Twisted Pair*) [64] el cual consta de 4 pares trenzados sin recubrimiento de categoría 3. Por lo que la frecuencia es de 25 MHz en longitudes de 100 m, esta distancia se puede aumentar mediante repetidores, siendo la máxima de 2500 m. En la imagen siguiente se muestra un conector RJ-45.



Número de pin	Señal	Descripción
Pin 1	TX+	Transmisión de datos
Pin 2	TX-	Transmisión de datos
Pin 3	RX+	Recepción de datos
Pin 4	BDD+	Dato bidireccional
Pin 5	BDD-	Dato bidireccional
Pin 6	RX-	Recepción de datos
Pin 7	BDD+	Dato bidireccional
Pin 8	BDD-	Dato bidireccional

Figura 37: Conector RJ-45

Tabla 15: Conector RJ-45

A continuación, se muestra el esquema de conexión entre el conector Ethernet y el sistema.

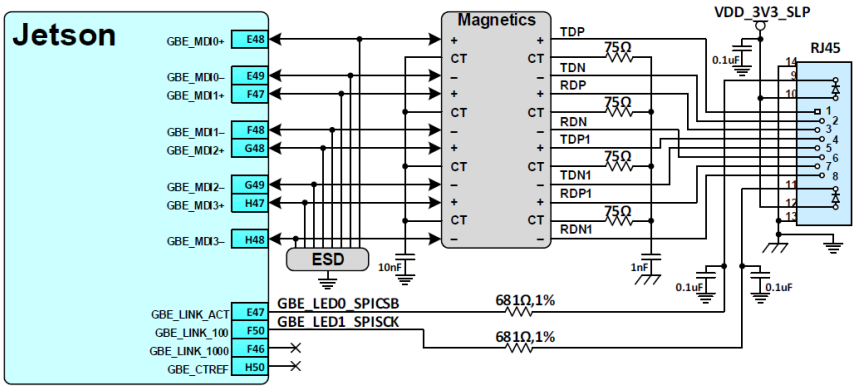


Figura 38: Conexión interna del RJ-45

4.1.13 Conector PCIe x4 pistas

Este bus se utiliza para conectar dispositivos periféricos directamente a la placa TX2, siendo esta versión Expres, por lo que la comunicación serie es mucho más rápida. Este conector está formado por cuatro carriles punto a punto, de carácter full-duplex y se trata de la versión 2.0 cuyas características se muestran a continuación.

- Línea de código de 8 o 10 bits.
- La velocidad de transferencia es de 5 GT/s.
- Ancho de banda por carril de 500 MB/s.
- Tamaño máximo de carga de 128 bytes
- Direccionamiento de 64 bits
- El tiempo de espera es configurable.
- No admite configuraciones TLP (*Transaction Load Packages*) ni ECRC (*Error Cyclic Redundancy Check*).
- Interrupciones señaladas mediante mensajes.

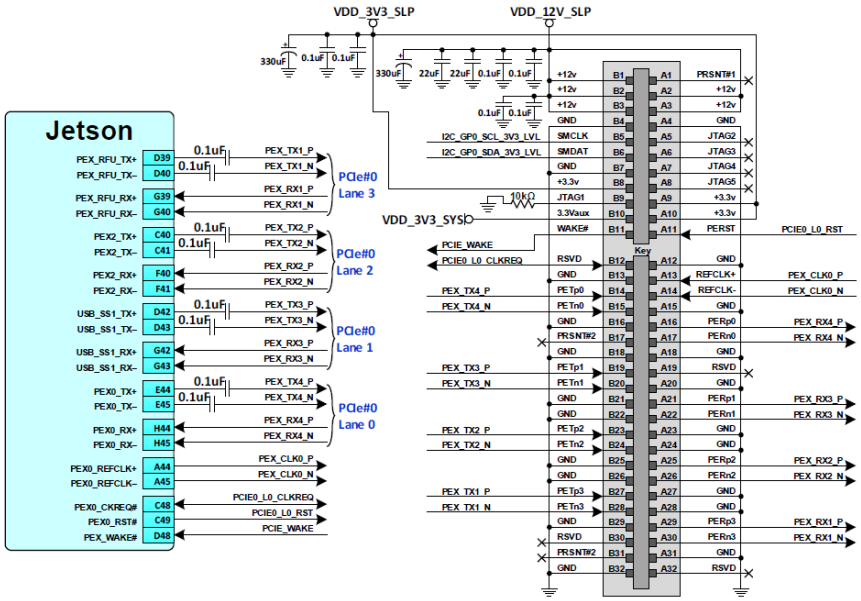


Figura 39: Conector PCI-E

4.1.14 Conector de expansión de pantalla

Este conector posee 120 pines distribuidos en 2 columnas, con 60 pines cada una. Su funcionamiento se basa en agrupar píxeles a la salida de los dos canales serie de visualización y codificarlos al formato seleccionado, posteriormente los transfiere a los dispositivos de salida, que suelen ser DP o HDMI. Admite ocho carriles de datos MIPI [40] DSI, lo que posibilita la conexión de dos interfaces de 4 carriles respectivamente, por ello se puede utilizar hasta dos pantallas de 4 carriles y un reloj para cada una. En el que cada carril de datos tiene un ancho de banda máximo de 1.5 Gb/s.

En la imagen siguiente se muestra la conexión del conector de pantalla DSI [22].

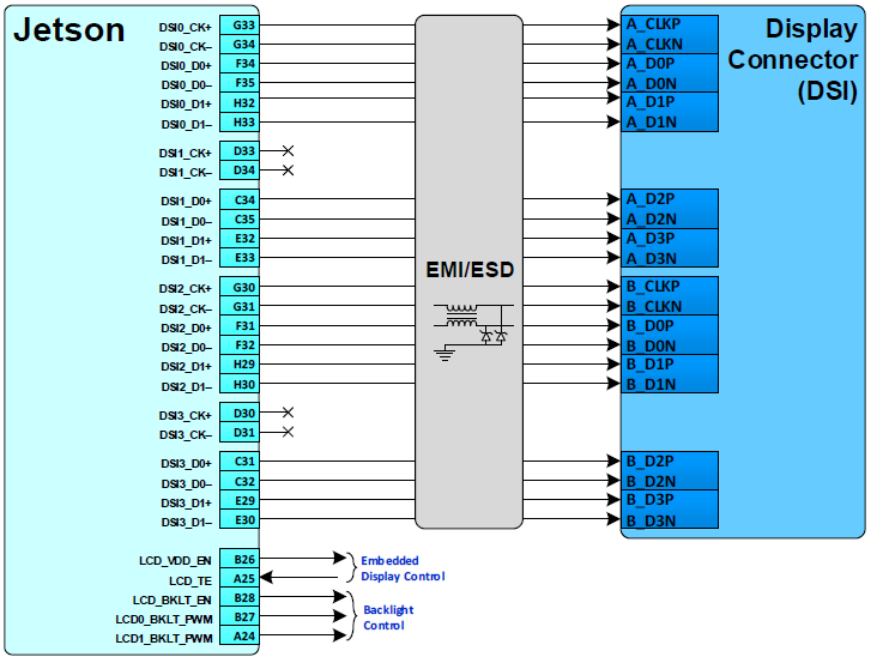


Figura 40: Conector pantalla DSI

Este conector posibilita la interfaz con los siguientes modos:

- DSI 2 x4: Conexión de dos pantallas de 4 carriles cada una.
- Estándar eDP: Utilizado en conexiones internas en pantallas de visualización.
- Estándar eDP AUX y HPD: Funcionan como un canal de banda lateral para la configuración del control de la pantalla.
- LCD BL EN/PWM: Conexión de pantalla LCD con habilitación de luz posterior y ancho de banda.
- Protocolo de comunicación SPI modo 0 y 2.
- I2C_GP1: Con una entrada de 3,3 V.
- INT / RST / CLK: Interrupción, control de reseteo y reloj de pantalla táctil.
- Control de pantalla

4.1.15 Conector GPIO

Este conector tiene 30 pines distribuidos en dos columnas de 15 pines cada una, que cuenta con un I2S y varios GPIO, estos últimos son pines genéricos cuyo comportamiento se puede controlar en tiempo real. Este conector puede ser utilizado como salida, entrada o interrupción, cuya conexión se realiza mediante el marco MPIO. El conector utiliza cinco tipos de pin MPIO, que son los siguientes:

- ST: Son los pines más comunes, se utilizan como entrada y salida del sistema.
- DD: Son pines de doble controlador, similares a los anteriores. Estos pines pueden soportar hasta 3.3 V si está en configuración de drenaje abierto.

- CZ: Estos pines se utilizan en aplicaciones que requieren de una salida de impedancia controlada.
- LV_CZ: Son similares a la anterior, pero en este caso se trata de un voltaje de 1.2 V.
- DP_AUX: este pin es utilizado como un canal auxiliar de control en el caso de que se requiera una señal diferencial de pantalla.

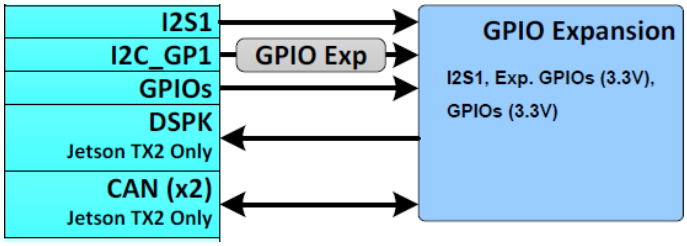


Figura 41: Expansión GPIO

Este conector admite la conexión a dos redes CAN, cuyo funcionamiento se basa en el modo bus para la transmisión de mensajes en entornos distribuidos.

Algunas características de este protocolo son:

- Modo FIFO programable.
- Modo de bucle programable para el autodiagnóstico, ya que es altamente inmune a las interferencias.
- Al tratarse de una red multiplexada, reduce el cableado y elimina la mayor parte de las conexiones punto a punto.

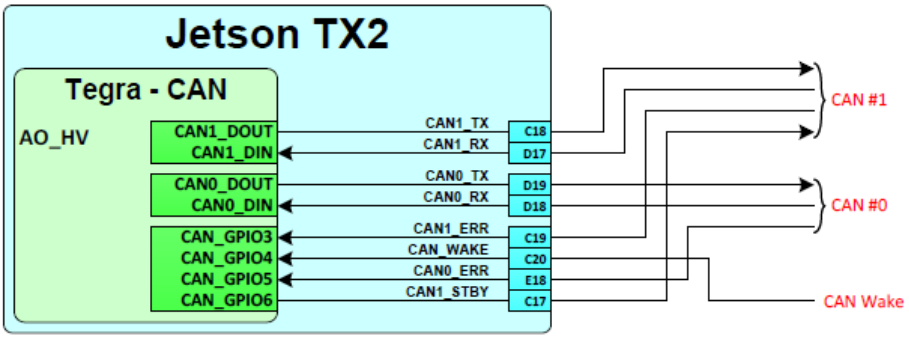


Figura 42: Conexión CAN

4.1.16 Conector de selección de voltaje

Como ya se ha observado algunas de las interfaces descritas requieren de 1.8 V o 3.3 V. Este conector de 3 pines se utiliza para controlar el cambio del nivel de voltaje requerido por los distintos interfaces.

Tiene dos modos de funcionamiento:

- Si los pines 1-2 del conector están en cortocircuito, se cambia el nivel de voltaje a 3.3 V.
- Por el contrario, si los pines 2-3 están en cortocircuito, el voltaje es 1.8 V.

En la siguiente tabla se muestran algunas de los módulos de la placa TX2 que requieren de 3,3 V y de 1,8 V.

Conector o módulo	Señal
I2C_GP0	3.3 V
I2C_GP1	
UART1_x_HDR	
Conector Ethernet	
Conector PCIe	
AUDIO_I2S	1.8 V
JTAG	
Tarjeta SD SDR104	

Tabla 16: Señales de conexión

4.1.17 Módulo de la cámara

Este conector tiene 120 pines distribuidos en 2 carriles de 60 cada uno y posibilita opciones de interfaz con varios módulos (UART, I2C), cámaras (CSI) y algunos sistemas de audio (I2S y DMIC).

MIPI Interfaz Serie de la Cámara (CSI)

Este estándar es compatible con las capas físicas D-PHY [19] v1.2 y C-PHY [10] v1.0, y posibilita la interfaz entre una cámara digital y un procesador. Esta tarjeta TX2 admite 3 bloques MIPI CSI, los cuales permiten la conexión de varios tipos de dispositivos y la configuración de la cámara. El ancho de banda es de 2,5 Gb/s y su funcionamiento consiste en introducir los datos a la pila FIFO asíncrona, la cual interactúa con el bloque NVCSI.

Sus principales características son:

- Intercalado de canales virtuales.
- Intercalado de tipos de datos.
- Procesamiento paralelo de píxeles, lo que aumenta el rendimiento y velocidades del reloj.

En la imagen inferior se muestra la interfaz CSI, su funcionamiento consiste en descargar instrucciones de procesamiento de audio y video desde el subsistema de la CPU, utilizando para ello un conjunto de bloques funcionales. Siendo de este modo más rápido y eficiente. A su vez está compuesto por diferentes subsistemas, que se muestran a continuación.

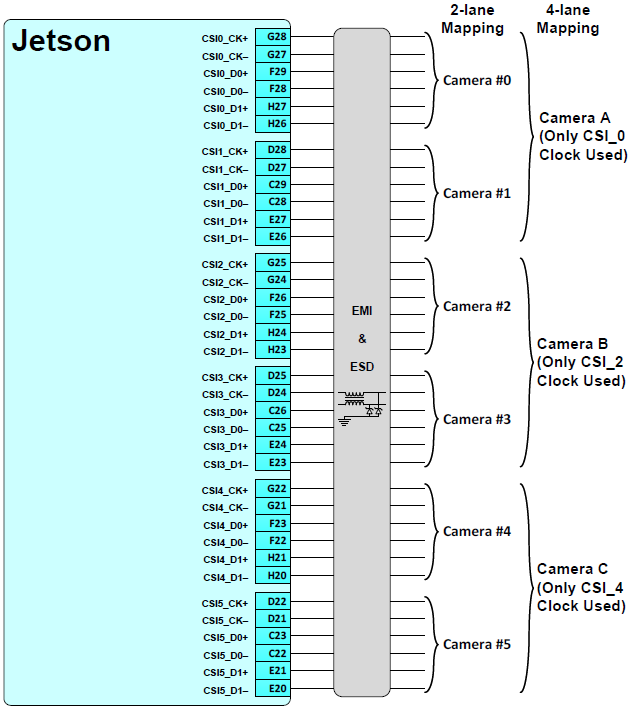


Figura 43: Interfaz CSI

Se observa que es posible la conexión de hasta 6 cámaras de doble carril. Cada bloque MIPI CSI tiene un reloj que se encarga del funcionamiento de un bloque de cámaras.

Decodificador de video multiestándar

Permite acelerar la decodificación de datos y admite resoluciones tanto bajas, estándar SD y altas HD [32] y UHD. Esta transferencia de datos se lleva a cabo mediante DMA (*Direct Memory Acces*) [17], por lo que no es necesario la intervención de la CPU. Puede funcionar a baja frecuencia en aquellas operaciones de potencia baja mediante un escalado de frecuencia. Además, es compatible con un gran número de estándares como H.265, VC-1 o WebM.

Codificador de video multiestandar

Este subsistema es el encargado de realizar la codificación de vídeos en alta calidad utilizado para realizar grabaciones. Además, permite la codificación de:

- La velocidad y resolución en una codificación simultánea múltiple secuencial.
- El control de frecuencia CBR (*Constant Bit Rate*) y VBR (*Variable Bit Rate*).
- Codificación de entropía: Es el mínimo número de bits por píxel necesarios para la codificación de una imagen, sin obtener pérdida de información.
- La sincronización de audios y vídeos.
- Cuantificación del post procesamiento.

Procesamiento JPEG

Este formato es el más utilizado por las cámaras y otros dispositivos, su funcionamiento consiste en un algoritmo que reduce el tamaño de los archivos de imágenes, comprimiéndolos por lo que al visualizarlo no se obtiene la imagen original. Además, realiza escalados de imágenes, decodificación (YUV420, YUV422H/V, YUV444, YUV400) y transformación del espacio de color (RGB a YUV). En la siguiente sección (4.2.4) se describe de manera breve en que consiste cada una de estas funciones.

Compositor de imágenes de video (VIC)

Este bloque procesa los datos recibidos del CSI y los transmite a la memoria del sistema o al procesador de señal de imagen (ISP). Es compatible con los formatos RGB, YCbCr [68] y datos crudos de Bayer. Las señales de entrada se obtienen de sensores CMOS [9] compatibles con MIPI y realiza diversas operaciones como con imágenes 2D, escalado, operaciones de mezcla, rotación, post procesamiento durante la reproducción de vídeos y la eliminación de ruidos. Una vez realizadas estas operaciones el procesador ISP recibe los datos del bloque VIC en formato Bayer y lo procesa al formato YUV.

4.1.18 Conector de expansión

Este conector está formado por 40 pines colocados en dos columnas con 20 cada uno y funciona como interfaz con diversos sistemas de audio y control como son el I2S, control de audio, micrófono digital IF o UART. Su formato es full-dúplex por lo que es bidireccional simultáneo y admite velocidades de reloj de hasta 49.152 MHz. Se le puede conectar cuatro salidas de audio PCM/I2S, y este procesamiento se realiza mediante el formato APE (*Monkey´s Audio*). Este utiliza un algoritmo para comprimir el audio sin producir pérdida de información y mantener su calidad original. La ventaja de este algoritmo es que no se produce pérdida de información y mediante el reloj de audio, se consigue procesar con potencia ultra baja. EL bus utilizado I2S es el encargado de interconectar circuitos de audio digital y separar las señales de datos y de reloj. Contiene mínimo tres líneas:

- Línea de reloj de bit: Da un pulso por cada bit discreto que se encuentre en la línea de datos. La frecuencia a la que opera es múltiple al valor de muestreo.
- Línea de selección de palabra: se utiliza para detectar si los datos enviados pertenecen al canal 1 o al 2, ya que dos canales pueden ser enviados por la misma línea de datos.
- Línea de datos: Formado por dos canales y es el medio por el que se envía el conjunto de datos.

La siguiente imagen muestra la conexión de los dispositivos audio con el módulo Tegra [58].

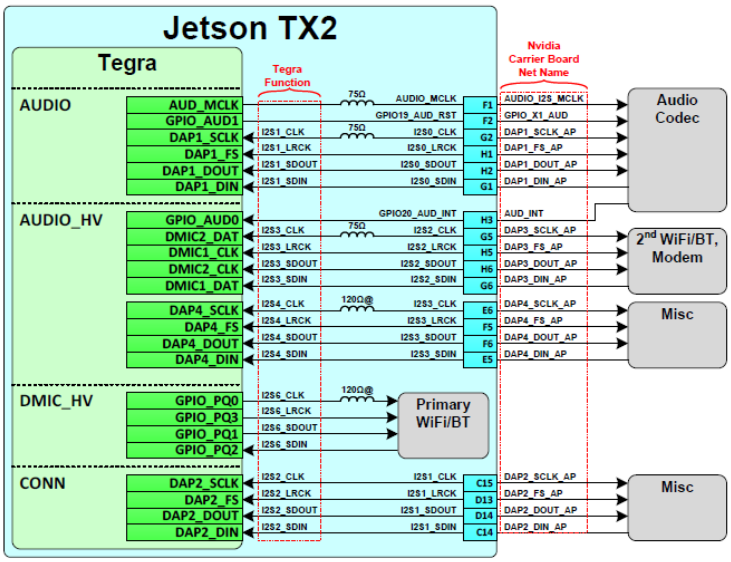


Figura 44: Conexión del audio

4.1.19 UART

La UART de la placa TX2 está conectada mediante dos bloques de cambio de nivel a un conector puerto serie de 6 pines. Este bloque cambia el voltaje de 1.8 V a 3.3 V, siendo este último al que está conectado el conector puerto serie. La UART permite la sincronización y conversión de datos, pasando de paralelo a serie y viceversa. Una de sus funciones es añadir bits de inicio para conseguir de este modo la sincronización del flujo de datos. Otra es la detección de errores, por ello añade un bit de paridad al carácter de datos. Este bit de paridad es verificado por el receptor cuando le llega el flujo de datos, para detectar si existe algún error en la transmisión. Como se observa en la imagen la UART controla el funcionamiento de la cámara.

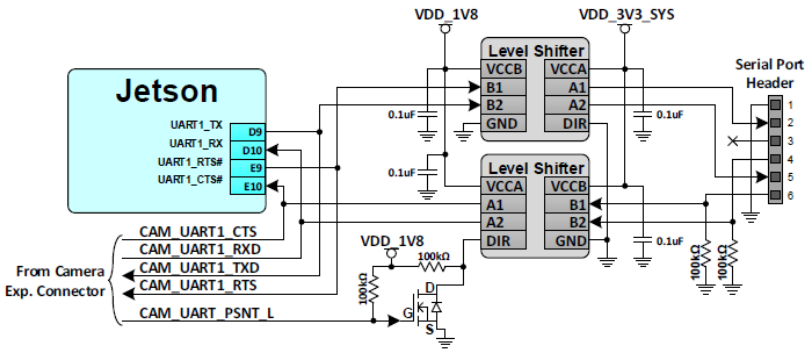


Figura 45: Conexión UART

Algunas de sus características más destacables son:

- Sincronización del flujo de datos y detección de errores con bit de inicio.
- La frecuencia del reloj puede llegar hasta 200 MHz.
- Las tramas de los datos pueden ser de 5 u 8 bits, en el que se incluye de manera opcional un bit de paridad y uno o dos bits de stop.
- Soporte para entradas de control de módem.
- Capacidad de acceso directo de memoria para TX y RX.
- Memoria FIFO [28] TX de 8 bits.
- Memoria FIFO RX de 11 bits, de los cuales 3 bits detectan los errores de ruptura, encuadre y errores de paridad.
- Detección automática de baudios.
- Realiza una interrupción en el caso de que la transmisión de entrada se detenga.
- Se puede determinar la prioridad de las interrupciones.
- Soporte de control de flujo a RTS y CTS.

4.1.20 JTAG

El módulo JTAG se encarga de testear y realizar pruebas de los sistemas conectados a él. Este se encuentra unido a un conector externo JTAG de 20 pines distribuidos en dos columnas, de los cuales nueve están conectados a masa. Se caracteriza por tener la conexión "*Daisy chain*", que permite la conexión sucesiva de varios dispositivos, permitiendo de este modo el uso de un único módulo JTAG para el sondeo de los diversos módulos.

Los pines de conexión del módulo JTAG son los siguientes:

- TDI: Entrada de Datos de Testeo.
- TDO: Salida de Datos de Testeo.
- TCK: Reloj de Testeo.
- TMS: Selector de Modo de Testeo.
- TRST: Reseteo de Testeo, es opcional.
- RTCK: Salida de reloj de prueba de retorno.

En este caso el modo de escaneo de límite no es posible, por lo que se ha de dejar desconectado el pin JTAG_GP0. Debido a ello se ha de realizar un puente entre el pin RESET_OUT y el conector de dos pines como se muestra en el esquema inferior.

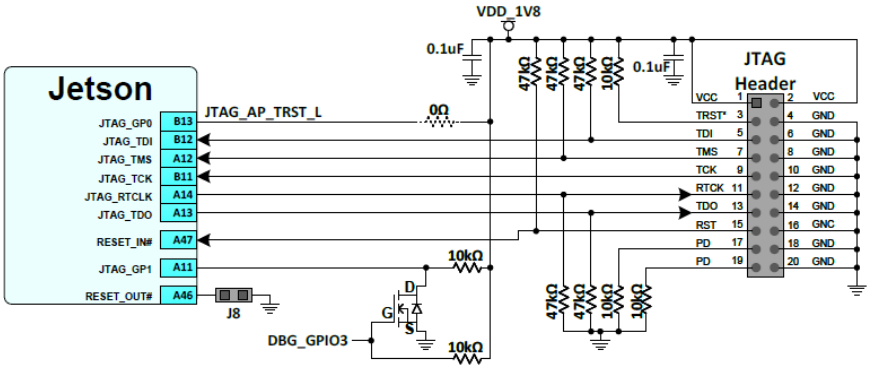


Figura 46: Conexión JTAG

4.1.21 Botón de encendido (BTN)

Este botón se utiliza para encender el sistema si está apagado, y ponerlo en funcionamiento o apagarlo en el caso de que este encendido. Para forzar un apagado del sistema se ha de mantener pulsado el botón en torno a 7 segundos.

4.1.22 Botón de recuperación (REC)

Se usa para activar el modo de recuperación, para ello se ha de mantener pulsado mientras el sistema se está encendiendo. Otro modo es mantenerlo presionado y pulsar el botón RST de reinicio.

4.1.23 Botón de Stop

Este botón se utiliza para poner el sistema en modo de suspensión, poniéndose de este modo en un consumo de potencia baja.

4.1.24 Botón de reseteo (RST)

Se usa para forzar un reinicio completo del sistema, es decir realizar un reseteo de toda la tarjeta TX2.

4.1.25 Conector SATA

Este interfaz utiliza un bus para la transferencia de datos entre la placa base y dispositivos de almacenamiento externo, pudiéndole conectar como por ejemplo un disco duro. El controlador es capaz de soportar el máximo rendimiento de un driver de segunda generación. Se caracteriza por proporcionar mayores velocidades y longitudes de cable de transmisión. Este módulo SATA corresponde a la versión 3.1 cuya velocidad de transmisión es de 6 GB/s, dando una velocidad máxima del disco de 600 MB/s de escritura y lectura de datos.

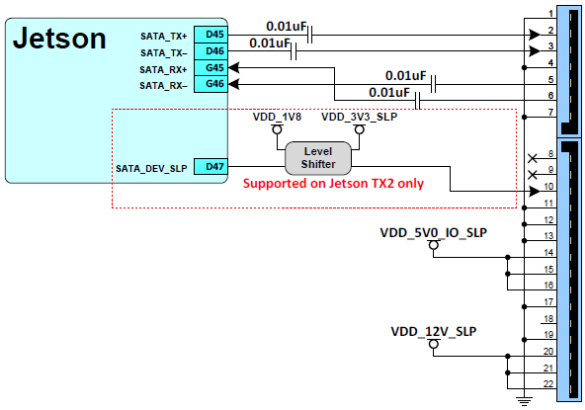


Figura 47: Conector SATA

Esta interfaz permite la conexión de dos conectores, uno de 7 pines y el otro de 15. La longitud máxima del cable es de 100 metros y sólo permite la unión entre la placa y una unidad externa.

Conector estándar de 7 pines

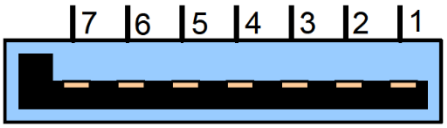


Figura 48: Conector SATA 7pines

Pin	Descripción
1	Masa
2	Señal de transmisión TX+
3	Señal de transmisión TX-
4	Masa
5	Señal de recepción RX-
6	Señal de recepción RX+
7	Masa

Tabla 17: Conector SATA 7 pines

Conector estándar de 15 pines

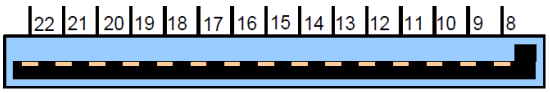


Figura 49: Conector SATA 15 pines

Pin	Descripción
8	No se utiliza
9	No se utiliza
10	Cambio de nivel de tensión
11	Masa
12	Masa
13	Masa
14	Tensión de 5 V
15	Tensión de 5 V
16	Tensión de 5 V
17	Masa
18	No se utiliza
19	Masa
20	Tensión de 12 V
21	Tensión de 12 V
22	Tensión de 12 V

Tabla 18: Conector SATA 15 pines

4.2. Conceptos de inteligencia artificial (IA)

El concepto de IA hace referencia a la inteligencia de las máquinas, las cuales a través de sensores son capaces de percibir su entorno y llevar a cabo una determinada acción. Por ello una de las tecnologías en las que se apoya es la visión artificial, ya que mediante esta disciplina las máquinas son capaces de adquirir, procesar o analizar su entorno. Las máquinas reciben una gran cantidad de información a partir de los sensores por lo que en la mayoría de las ocasiones esta debe ser seleccionada, en el caso de que la fuente de información sea una cámara se aplica el procesamiento de imágenes.

Para considerar que una máquina posee inteligencia esta debe de ser capaz de analizar y clasificar la información que recibe y ante esto generar una respuesta o acción similar a la que generaría el ser humano. Por lo que una de las herramientas que utiliza son las redes neuronales, dotándole así de una inteligencia artificial, que se ha conseguido mediante un entrenamiento previo de la máquina.

4.2.1 Redes Neuronales Artificiales (RNA)

El análisis de las neuronas se remonta a la década de los cuarenta, cuando en 1943 McCulloch y Pitts crean un modelo abstracto de la neurona. En el que la probabilidad de que una neurona tenga de activarse depende de la señal de entrada y de la sinapsis de la conexión, describiendo así la actividad neuronal mediante un álgebra booleano. (Redes neuronales artificiales fundamentos y aplicaciones, 1993) [81]

Perceptrón

La unidad central de la red recibe el nombre de neurona y al realizar su estudio se puede considerar que está formada por la unión de dos nodos como se muestra en la siguiente imagen.

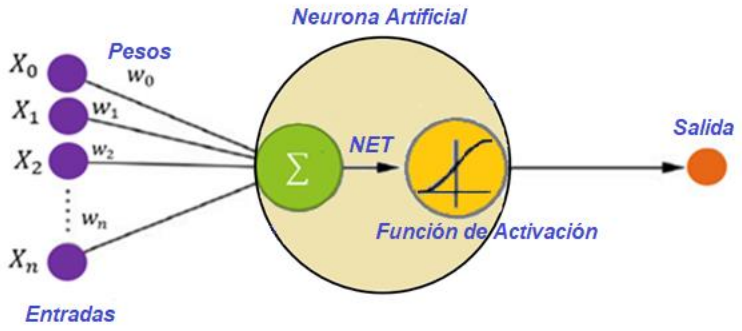
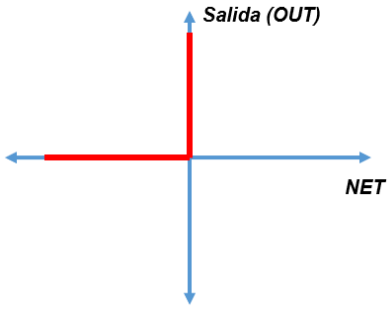


Figura 50: Partes de una neurona

Como se puede observar el término NET corresponde a la suma de todas las combinaciones de entradas con sus respectivos pesos, siendo estos últimos los que determinan el efecto cuantitativo de las entradas.

Función de activación

A la entrada NET se le aplica una función de activación generando de este modo una señal de salida, a la que se le denomina ganancia. Esta es proporcional a la pendiente de la curva de la función según el valor de NET. Si no se ajusta el valor de esta ganancia puede llegar a saturar el sistema, por lo que se le ha de aplicar una función de activación que ajuste lo máximo posible el valor de esta ganancia. En la siguiente imagen se muestra la salida que presenta la función NET sin aplicarle ninguna función de activación.



En la gráfica se observa que, si el valor NET es negativo, la ganancia es nula, mientras que a partir de cero la ganancia se incrementa, llegando a saturar el sistema. Para solucionar esto se aplican funciones de activación entre las que destacan la sigmoide, tangente hiperbólica y relu.

- Sigmoide

Con el uso de esta función se consigue que para valores NET muy pequeños el valor de la ganancia sea alto, mientras que, para valores muy altos la ganancia sea nula. Se observa en la gráfica que en la zona intermedia hay una elevada pendiente, mientras que en los extremos es casi horizontal. Al ser su derivada simple se suele aplicar en la técnica del descenso del gradiente. Su expresión matemática es:

$$Salida(OUT) = \frac{1}{1 + e^{-NET}}$$

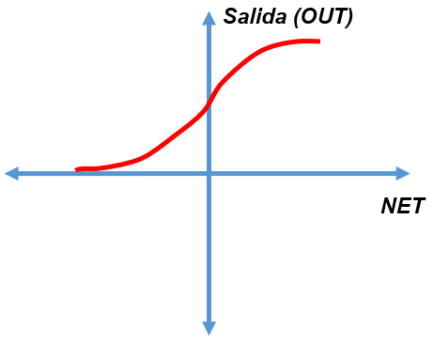


Figura 51: Función Sigmoide

- Tangente hiperbólica

Es similar a la anterior, pero a diferencia de esta se pueden utilizar mayor cantidad de datos de entrada y puede dar como resultado valores negativos. Su expresión matemática es la siguiente:

$$Salida(OUT) = \frac{e^{NET} - e^{-NET}}{e^{NET} + e^{-NET}}$$

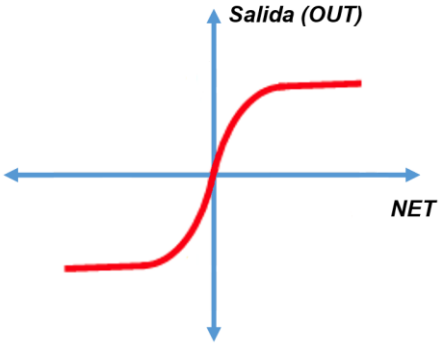


Figura 52: Función Tangente hiperbólica

- Relu

Esta función de activación se utiliza principalmente en redes convolucionales, ya que los resultados obtenidos tienen menor probabilidad de error. La expresión matemática que representa esta función es:

$$Salida(OUT) = \max(0, NET)$$

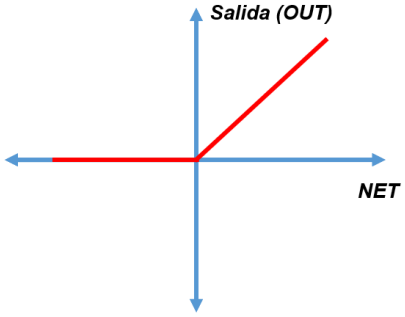


Figura 53: Función RELU

Perceptrón Multicapa (MLP)

Al conectar varias neuronas se crea la red neuronal la cual está organizada en capas, estas se pueden dividir en tres tipos según la zona donde se encuentren:

- Capa de entrada: Es mediante la cual los datos son presentados a la red.
- Capa intermedia: Es el conjunto de capas que se encuentran entre la capa de entrada y salida.

- Capa de salida: Es la que muestra el resultado obtenido de las entradas tras pasar todas las capas.

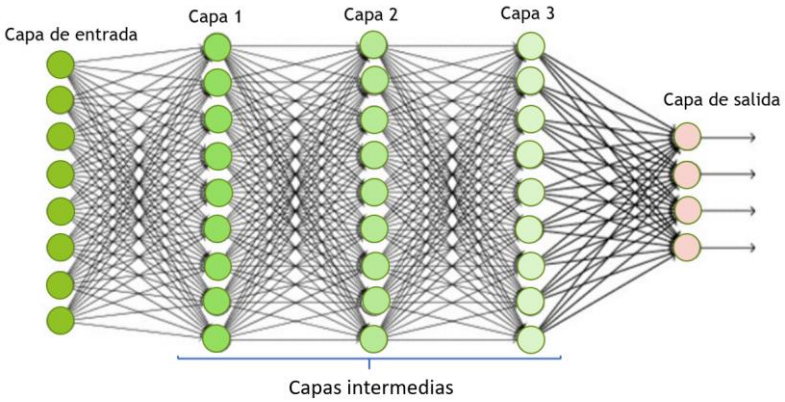


Figura 54: Capas de una red neuronal

A la hora de determinar la salida del conjunto de toda la red depende de la función de activación que estas tengan. En el caso de que la función sea lineal se obtiene la salida como la multiplicación del número de entradas por la matriz W.

$$Salida = X_n \cdot W$$

Siendo W la matriz formada por el conjunto de pesos de la RNA, cuyas dimensiones son filas (entradas) por columnas (neuronas). Por el contrario, si la función de activación es no lineal, no se le puede aplicar el producto de las matrices, sino algoritmos especiales como la retropropagación.

Topología

Hace referencia a la estructura de conexión que presentan las neuronas de la red, las cuales se pueden clasificar en tres tipos.

Feedforward

Es la configuración más simple, su modo de conexión es lineal y las neuronas se conectan con la siguiente capa hasta llegar a la capa de salida. Por lo que las neuronas de cada capa solo dependen de la capa anterior y no existen conexiones entre las neuronas de una misma capa.

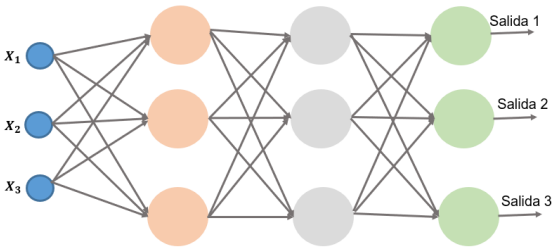


Figura 55: Estructura Feedforward

Elman o recurrentes simples

En esta estructura las salidas de las neuronas son retroalimentadas sólo a la capa anterior más cercana, por lo que la reducción de error puede reducirse en menos número de iteraciones que a diferencia de la estructura anterior. Esto se debe a que la retroalimentación contiene un retraso lo que permite a las neuronas conocer el valor obtenido previamente y ajustar los valores de los pesos reduciendo así el error. Sin embargo, esto supone un mayor tiempo de procesamiento, por lo que en redes donde hay un gran número de neuronas o capas, es muy lento el entrenamiento de estas redes.

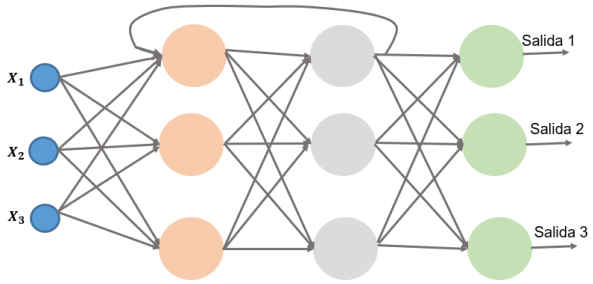


Figura 56: Estructura Elman

Feedback o recurrentes completas

Este modo de conexión tiene una estructura cíclica, en el que las neuronas se pueden conectar con ellas mismas, con neuronas de capas anteriores o siguientes. Esto le otorga a la red un carácter temporal, en el que el valor itera durante un tiempo hasta alcanzar un criterio de convergencia, por lo que la red tiene un comportamiento variable en el tiempo.

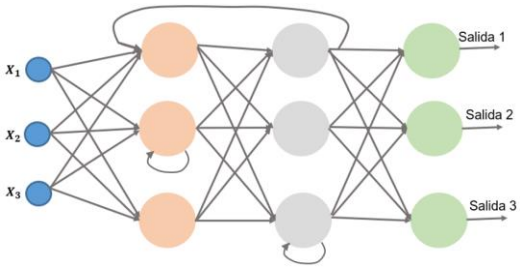


Figura 57: Estructura Feedback

Aprendizaje

Se le denomina al proceso en el que las neuronas actualizan los valores de los pesos, ya que el producto de estos con el valor de la entrada es la información utilizada en el aprendizaje. Esto se debe a que los pesos representan el estado actual de conocimiento y son distribuidos al resto de neuronas, por lo que se considera que la red tiene una memoria distribuida. El aprendizaje se lleva a cabo mediante reglas que cambian los pesos en respuesta a las entradas dadas y existen varios tipos según como se lleva a cabo este.

Supervisado

Se establece las entradas y las salidas deseadas, por lo que mediante un modelo neuronal basado en la observación de ejemplos se ha de relacionar ambos parámetros. Cuando el entrenamiento se inicia los pesos son determinados de manera aleatoria, cuyos valores son pequeños lo que evita que la ganancia sature el sistema, generando así con un valor de error aleatorio. El entrenamiento comienza por la capa más cercana a la entrada en la que se determina el valor NET en cada neurona y se le aplica la función de activación correspondiente, generando así una salida que pasa a la siguiente capa como entrada. Este proceso se repite en todas las capas hasta llegar a la salida de la red, donde se compara el valor alcanzado con la salida deseada, determinando así el error obtenido.

El objetivo del aprendizaje es establecer una determinada estructura de la red en la que el error de salida se minimice lo máximo posible. Por ello como el valor del error obtenido es proporcional al conjunto de pesos establecidos W, se ha de determinar los pesos para la próxima iteración. Por lo que se realiza la derivada de la función del error con respecto a los pesos determinando así el valor del gradiente. Con estos valores se realiza la propagación hacía atrás, en las que cada neurona recibe la fracción de la señal de error en función de la contribución que esta hubiera hecho, de este modo se determinan los nuevos valores de los pesos de cada neurona.

No supervisado

En este no se determina una salida deseada, por lo que la RNA aprende de manera autónoma. Mediante este aprendizaje se pretende que la red descubra las características y correlaciones existente en los datos de entradas, y los categorice. Uno de los modelos que se utiliza en este aprendizaje es *El mapa autoorganizativo de Kohonen.* Qu consiste en presentar un gran número de entradas que tengan algún parecido o relación, ya que en caso contrario no se podrá realizar un entrenamiento eficiente. Se determina el producto de las entradas con los pesos y se elige aquel cuyo valor sea más cercano a la señal de entrada. La neurona a la que le corresponde este valor es tomada como referencia a la cual se le da un valor de uno y las de su entorno van decreciendo en función de la cercanía a la que se encuentran de esta. Además, el valor obtenido del producto de los vectores de entradas y pesos ha de ser reforzado para que de este modo en la próxima iteración la red responda de manera similar ante entradas parecidas.

Error de entrenamiento

Al realizar el entrenamiento se puede dar el caso de que la red se sobreentrene, por lo que el modelo se ajusta a los datos con los que ha sido entrenada, dando sólo resultados válidos aquellos ejemplos que se ajusten a los datos entrenados (Caparrini and Work,2018) [8]. Esto se debe a que el modelo de la red es muy complejo en relación con los datos de entrada, por lo que no tiene los suficientes datos para capturar el modelo de datos subyacentes y pierde su capacidad de generalizar y relacionar nuevos ejemplos.

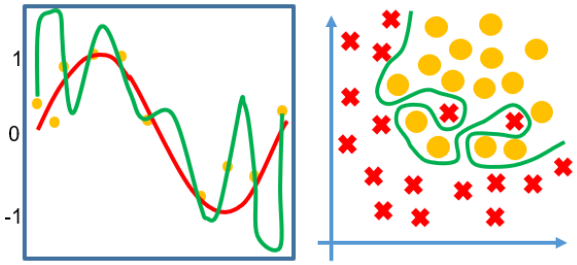


Figura 58: Sobreentrenamiento

Otro error que se puede obtener en el entrenamiento es que sean pocas las diferencias que existan entre los datos de entrada, generando de este modo un modelo simple. Por lo que no hace una buena diferenciación entre ellas y ante un nuevo ejemplo le resulta difícil predecir con qué tipo de los ejemplos se relaciona más.

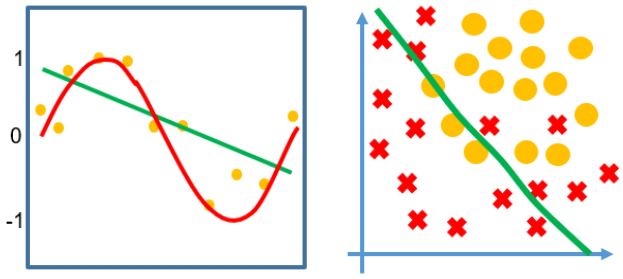


Figura 59: Entrenamineto poco efectivo

La siguiente imagen muestra un modelo de entrenamiento ajustado y válido.

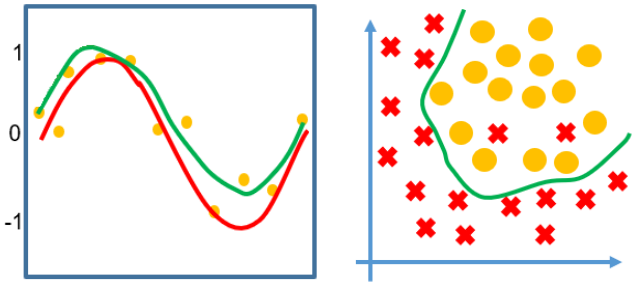


Figura 60: Entrenamiento ideal

Datos utilizados

Los datos con los que se trabajan en las redes neuronales se dividen en dos bloques que es entrenamiento y prueba.

- Entrenamiento: son los utilizados para determinar los pesos sinápticos y ajustar de este modo los parámetros. La matriz de estos pesos depende del número de entradas y del número de neuronas.
- Prueba: Una vez realizado el entrenamiento se utiliza un conjunto de datos que no habían sido utilizados anteriormente, para validar el aprendizaje. Se deben probar aquellos datos que abarquen todos los casos posibles y pudiera dar a confusión. De este modo se le plantea los casos más complejos y se comprueba la eficiencia del entrenamiento de la red.

Redes Neuronales Convolucionales CNN

Es un tipo de RNA, cuyo funcionamiento es similar pero su estructura está formada por diferentes tipos de capas. Estas se caracterizan por trabajar con pequeñas cantidades de datos, cuya información combina en sus capas internas, por lo que se utiliza cuando el número de entradas es excesivo como es el caso de las imágenes.

Capa convolucional

Esta capa a diferencia de una capa normal utiliza filtros en vez de realizar la multiplicación de las entradas por los pesos. Por lo que se obtiene como resultado una imagen con aquellas características destacadas por la máscara o filtro utilizado, reduciendo así el número de cálculos y parámetros.

Capa pooling o de reducción

Está colocada en medio de dos capas y se encarga de reducir el tamaño de la imagen obtenida de la capa anterior. El uso de esta capa implica una pérdida de información, lo que se considera como una ventaja, ya que supone la reducción de cálculos y el ajuste de parámetros o datos. Para realizar la reducción del tamaño de la imagen se realizan diferentes operaciones siendo la más utilizada la max-pooling. Su funcionamiento consiste en dividir la imagen de entrada en regiones más pequeñas, de las cuales coge el rectángulo de máximo valor de cada una.

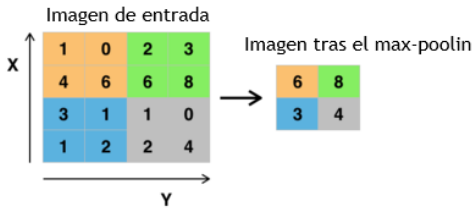


Figura 61: Función Max-pooling

Capas conectadas totalmente

Es la última capa de la red neuronal en la que cada píxel se considera como una neurona independiente. Por lo que habrá tantas neuronas como el número total de casos que se deba predecir.

4.2.2 Tensor

Hace referencia al conjunto de datos que forman una matriz de N dimensiones, cuyos valores pueden ser transformados mediante filtros o kernels. Por ejemplo, el tensor de una imagen de resolución 416x416 es:

$$Tensor = 416x416x3$$

Donde 416x416 hace referencia al ancho y alto de la imagen, y el 3 a la profundidad de los colores RGB. La siguiente imagen muestra los diferentes tipos de tensores que existen según las dimensiones de este.

- 1d-Tensor: Es un vector el cual es de una dimensión.
- 2d-Tensor: Matriz de dos dimensiones.
- 3d-Tensor: Es un cubo por lo que tiene tres dimensiones.
- 4d-Tensor: Hace referencia a un vector de cubos.
- 5d-Tensor: Es una matriz formada por N^2 cubos.
- 6d-Tensor: Cubo compuesto por N^3 cubos internos.

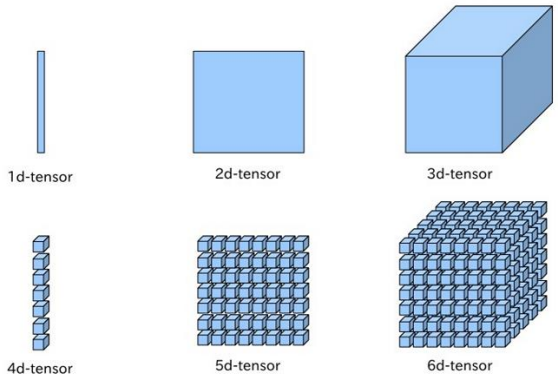


Figura 62: Tipos de Tensores

4.2.3 Deep Learning vs Machine Learning

Son conceptos relacionados pero diferentes, ambos se basan en la inteligencia artificial pero solo las aplicaciones con Deep Learning son capaces de entrenarse ante nuevos datos de entradas. Mientras que las denominadas Machine Learning aprenden y clasifican los datos de entrada y ante una nueva entrada busca la relación existente con los datos aprendidos.

Machine Learning

Su modo de funcionamiento consiste en entrenar a la red neuronal con un conjunto grande de datos, siendo capaz de aprender y clasificar esta información. Se basa principalmente en un entrenamiento supervisado en el que la persona le indica cual ha de ser el resultado esperado. Esto supone una gran carga de trabajo, ya que la persona previamente ha de etiquetar cada una

de las entradas, para que la red aprenda estas características y posteriormente sea capaz de detectar los objetos. Actualmente este entrenamiento lo puede llevar a cabo cualquier programador que cuente con las herramientas necesarias, ya que existen diversos algoritmos que están disponibles por la red. Además, se pueden descargar librerías de conjuntos de imágenes ya etiquetadas o redes ya preentrenadas.

Deep Learning

A diferencia del anterior en este se integra el entrenamiento de carácter no supervisado en el que no es necesario que el ser humano intervenga en la toma de decisión. Este modo de aprendizaje se acerca más al que realizan las personas, en el que los algoritmos son más complejos y los modelos utilizados en el entrenamiento intentan imitar el sistema nervioso del ser humano. Se caracteriza por utilizar múltiples capas en el que algunas de estas se especializan en detectar determinadas características ocultas en los datos. (Arrabales, 2018) [6]

Uno de los principales avances que se dio en este entorno fue el reconocimiento de voz que se consiguió mediante el aprendizaje LSTM (*Long Short Term Memory*). Este está basado en el entrenamiento de redes neuronales recurrentes con lo que se evita que el gradiente se desvanezca en su propagación hacía atrás (En.wikipedia.org, 2018) [21]. Se toma como base una neurona a la que se le añade un mecanismo de olvido, mediante el cual con una nueva entrada este sepa que características le vale la pena seguir conservando u olvidar. Se le introduce un mecanismo de guardado que es quién decide qué grado de la información de entrada es válida y uno de salida, mediante el cual se indica que parte de la información de la memoria de largo plazo es útil, deshaciéndose de aquella información irrelevante. (Blog.echen.me, 2018) [7]

Cada neurona está formada por:

- Una celda principal: Es donde almacena temporalmente la información de las entradas.
- Una puerta de entrada: Es la encargada de controlar en que grado la nueva entrada influye en la celda principal.
- Puerta de olvido: Esta decide qué información vale la pena almacenar u olvidar.
- Una puerta de salida: Determina que parte de la salida de la celda es utilizada como salida de la LSTM, quedándose en la celda almacenada solo la información que considera interesante.

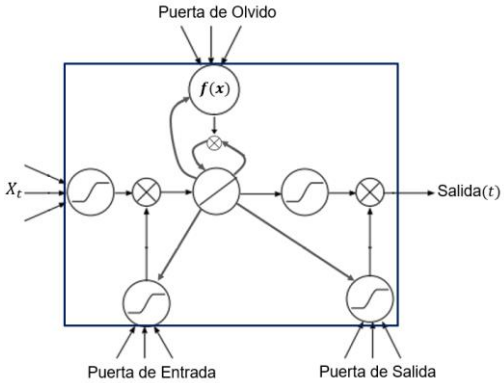


Figura 63: Red LSTM

Algunas de las aplicaciones que se están estudiando o llevando a cabo en base a Deep Learning es el coche autónomo, la traducción automática, el reconocimiento de caras u objetos en imágenes, la generación automática de subtítulos y otros sectores como la sanidad, la química y el empresarial. (Medium, 2018) [57]

4.2.4 Procesamiento de imágenes

Una imagen se puede analizar como una matriz de coordenadas (x, y) cuyo valor varía en función de la intensidad de la luz que presente la imagen.

$$I = f(x, y)$$

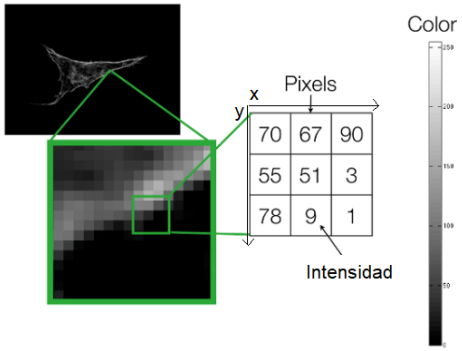


Figura 64: Representación de una imagen

Cada uno de los cuadrados en los que queda dividida la imagen se denomina píxel el cual se representa generalmente por 8 bits, lo que da lugar a una gama de 256 colores. La intensidad es representada por un valor comprendido entre el 0 y 255, siendo este último el color blanco y el 0 el negro. La forma de procesar una imagen en color y una en tono de grises es diferente, ya que si se trata de una imagen en escalas de grises sólo se requiere de un canal y una matriz. Mientras que si se trata de una imagen en color se requiere de tres canales RGB y por lo tanto la matriz es tridimensional.

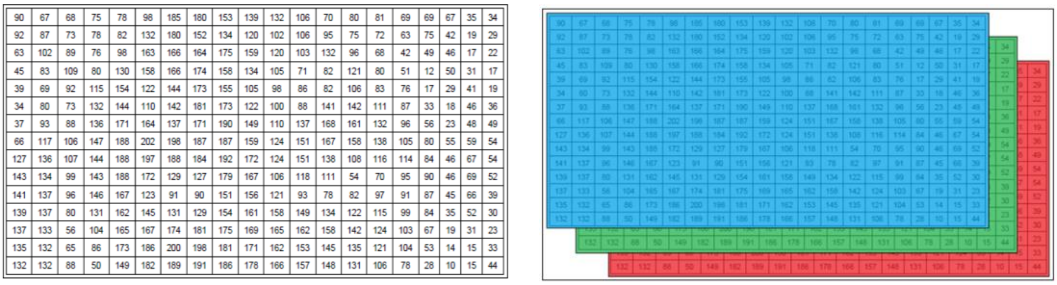


Figura 65: Matrices escalas de grises y RGB

Se pueden utilizar más canales para el almacenamiento de otros datos de interés como puede ser el nivel de opacidad de una imagen. El procesamiento de una imagen consiste en someter al conjunto de sus píxeles de la matriz a una transformación morfológico. Cuando la imagen es sometida a un conjunto de procesamientos morfológicos se le denomina filtrado, que se utiliza

para mejorar o resaltar alguna característica de la imagen. A continuación, se explicarán algunos ejemplos de estos procesamientos morfológicos y filtros.

Procesamiento Morfológico

Consiste en tomar los pixeles que se encuentran alrededor del píxel del que se desea calcular su valor, y crear una matriz del mismo tamaño que la matriz de convolución. Se multiplican ambas matrices y el resultado obtenido es el valor del píxel deseado, que suele estar ubicado en el centro de la matriz, al que se le denomina píxel de anclaje. Dependiendo de qué transformación morfológica se desee realizar a la imagen se realizará de un determinado modo y se utilizará una matriz de convolución o kernel explícita. La dilatación y erosión son dos operadores morfológicos básicos, y en función de cómo se combinen estos dan lugar a otras variantes como es la apertura y cierre.

Dilatación

El kernel se desplaza sobre la imagen original y con que al menos uno de los píxeles de la matriz de convolución se superponga en uno de la imagen, se considera que el píxel central del kernel es un uno, aumentando de este modo la región de la imagen.

$$Dilatación = A \oplus B$$

Figura 66: Dilatación

Erosión

En este caso si los pixeles del kernel al deslizarse sobre la imagen coinciden totalmente con estos se considera que el píxel de anclaje es un uno, por lo que no erosiona la imagen. Por el contrario, si los pixeles del kernel no se superponen en su totalidad sobre los pixeles de la imagen, el píxel de anclaje es un cero, erosionando de este modo la imagen. Esto sucederá con todos los píxeles cercanos al borde de la imagen.

$$Erosión = A \ominus B$$

Figura 67: Erosión

Apertura

Esta transformación morfológica se obtiene mediante la combinación de la erosión y dilatación. Se ha de tener en cuenta que el orden en el que se aplican los operadores morfológicos es muy importante, por lo que la imagen se somete primero a una erosión y posteriormente a la dilatación. Este método se utiliza especialmente para eliminar ruidos, suavizar contornos y separar formas con uniones muy finas.

$$Apertura = A \circ B = (A \ominus B) \oplus B$$

Figura 68: Apertura

Cierre

A diferencia del anterior en este método primero se realiza la dilatación y posteriormente la erosión. Se suele utilizar para la unión de dos zonas muy próximas en las imágenes o para cerrar pequeños huecos.

$$Cierre = A \cdot B = (A \oplus B) \ominus B$$

Figura 69: Cierre

Gradiente Morfológico

Este método da como resultado la diferencia existente entre la dilatación y la erosión, por lo que el resultado es el contorno de la imagen.

$$Grad.Morfológico = (A \oplus B) - (A \ominus B)$$

Figura 70: Gradiente Morfológico

Sombrero de Copa

Este método muestra la diferencia entre la imagen original y la de apertura.

$$Som.Copa = A - (A \circ B)$$

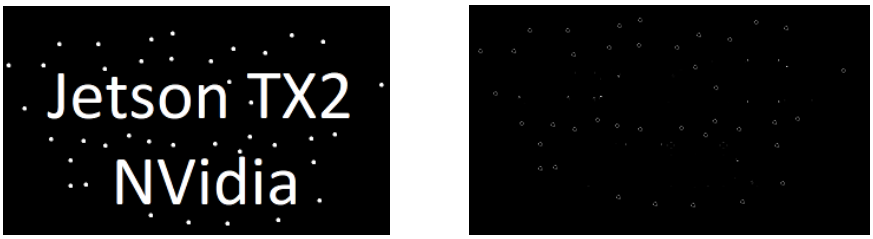


Figura 71: Sombrero de Copa

Sombrero Negro

Da como resultado la diferencia existente entre la imagen original y la de cierre.

$$Som.Negro = A - (A \cdot B)$$

Figura 72: Sombrero Negro

Filtros

La combinación de operaciones morfológicas da lugar a diferentes filtros, algunos de ellos son los siguientes.

Segmentación

Consiste en modificar la imagen original obteniendo otra imagen con características más significativas y fáciles de analizar. Ya que la imagen resultante está dividida en regiones, donde se encuentran conjuntos de píxeles que presentan características muy similares. Uno de los métodos utilizados en la segmentación es el del valor umbral, en el cual en función del valor establecido como umbral los píxeles se clasificarán en zonas, creando nuevas imágenes a su salida. Hay dos tipos:

- Umbral único: Se determina un único valor, por lo que si el valor del píxel supera el umbral formará parte del objeto.

Figura 73: Segmentación Umbral único

- Umbral multinivel: En él se determinan diferentes umbrales, por lo que cada píxel formará parte del objeto que se cree entre dos valores de los umbrales establecidos.

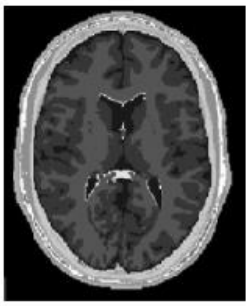


Figura 74: Segmentación Umbral Multinivel

Detección de contornos

Cuando en una imagen hay bordes resulta fácil detectarlo, ya que la intensidad de su píxel cambia de modo muy evidente. Por ello se utiliza la primera derivada o gradiente, la cual expresa el cambio de la intensidad del píxel, siendo el de mayor diferencia de intensidad el máximo de la función.

- Sobel: este operador combina suavizado y diferenciación Gaussiana, y el valor del píxel se obtiene de:

$$G = \sqrt{G_x^2 + G_y^2}$$

  Siendo G_x el resultado del cambio horizontal obtenido mediante la multiplicación de un kernel con la imagen original. Para obtener los cambios horizontales la columna central del kernel ha de estar compuesta de ceros. Por el contrario, para calcular los cambios verticales los números de la fila central del kernel han de ser ceros. Este se ha de multiplicar por la imagen original, obteniendo de este modo a G_y.

- Scharr: su funcionamiento es similar a Sobel, pero es más preciso. Se utiliza cuando el kernel es 3x3 y presenta la siguiente estructura.

$$G_x = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \qquad G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix}$$

- Canny: es el operador más preciso para detectar los bordes, ya que sus tasas de errores son muy bajas y localiza muy bien los bordes. Esto se debe a que realiza más operaciones y por lo tanto los resultados son más precisos. Lo primero que realiza es un filtrado gaussiano, en el que elimina cualquier tipo de ruido. Después se determina el gradiente de intensidad mediante el cálculo de G_x y G_y, y posteriormente se determina la fuerza y dirección del degradado. La fuerza se determina del mismo modo que en Sobel, mientras que la dirección se obtiene mediante la siguiente función.

$$\theta = arctan\left(\frac{G_y}{G_x}\right)$$

  El resultado obtenido de esta operación es redondeado a 0, 45, 90 o 135, que son los cuatro ángulos posibles. Después se le aplica una supresión no máxima, con la que se elimina aquellos píxeles que no forman parte del borde. Por último, los píxeles son sometido a histéresis, en los que se determina dos umbrales. Pueden darse tres casos, si el píxel pasa el umbral superior se considera como un píxel positivo. En el caso de que el valor del píxel se encuentre por debajo del umbral inferior, el píxel es eliminado. Por último, si el valor se encuentra entre los dos umbrales se ha de analizar si esta junto a un píxel positivo, si es así se acepta, en caso contrario es eliminado. Los valores de los umbrales máximos suelen ser 2 y 3, mientras que el mínimo es 1.

Suavizado de imágenes

Este filtro lo que pretende es reducir el valor de la variación de intensidad que hay entre los píxeles contiguos. Es similar a un filtro paso bajo en el que sólo afecta a los píxeles de mayor cambio de intensidad, por lo que es utilizado para eliminar pequeños detalles y ruidos.

- Difuminar: en este método el píxel que se quiere calcular se obtiene de la multiplicación de sus píxeles de alrededor por una matriz de convolución, cuyos valores tienen el mismo peso. El dato obtenido es dividido por el resultado de la multiplicación del kernel de la anchura y altura de la imagen.

- Filtro gaussiano: en este caso el valor del píxel se obtiene al promediar los píxeles de los lados con distintos pesos (σ), obteniendo de este modo una campana de Gauss. Se ha de tener en cuenta que cuanto mayor es el valor de σ, mayor ruido eliminará, pero se obtiene una imagen más difuminada. Con la siguiente ecuación se obtiene el valor del píxel, donde (i, j) representa las coordenadas del píxel a calcular y (k, l) la de uno a su lado.

$$g(x,y) = e^{-\frac{(i-k)^2+(j-l)^2}{2\cdot\sigma^2}}$$

  El píxel central es el de mayor valor y corresponde a la punta de la campana de Gauss. Los resultados obtenidos varían entre el cero y el uno, por lo que, para obtener una matriz de números enteros, esta se ha de dividir por el valor más pequeño obtenido.

- Filtro medio: su método para calcular el valor del píxel se basa en hacer la media de los píxeles de su alrededor. Aunque es muy fácil de utilizar tiene el inconveniente de que puede generar nuevos valores de intensidades en los píxeles que antes no aparecían en la imagen original.

- Filtro Bilateral: este método se utiliza para preservar los bordes de las imágenes y es similar al filtro gaussiano, pero en este caso se utilizan dos valores de σ. Con uno de ellos se realiza la misma operación que con el filtro gaussiano y el segundo representa la diferencia de intensidad entre el píxel calculado y los de su alrededor. La siguiente ecuación representa este cálculo, donde la I es la imagen original.

$$g(x,y) = e^{-\frac{(i-k)^2+(j-l)^2}{2\cdot\sigma^2}-\frac{|I(i,j)-I(k,l)|^2}{2\cdot\sigma^2}}$$

Escalado de imágenes

Es un procedimiento que cambia la resolución de la imagen, consiguiendo de este modo una mayor calidad con las mínimas pérdidas.

- Si se trata de una imagen gráfica vectorial el escalado se realiza mediante transformaciones geométricas, la cual no genera pérdida de calidad.
- Si es una imagen de gráficos por puntos, en este caso se genera una nueva imagen con un mayor o menor número de píxeles. En el caso de que se reduzcan el número de píxeles, esto genera una pérdida de calidad.

Decodificación

Este método decodifica la información de la matriz y pone cada valor en su casilla correspondiente. Posteriormente se multiplica cada uno de estos valores por el valor correspondiente de la matriz de cuantificación utilizada. Después se deshace la transformación discreta de coseno (DCT), y se le suma 128 a cada entrada. La transformación DCT consiste en convertir la señal al dominio de la frecuencia, dividiendo la imagen en pequeños bloques de 8x8 píxeles, que se procesan y se le resta 128 a cada entrada. Consiguiendo de este modo expresar una secuencia finita de varios puntos como resultado de la suma de distintas señales sinusoidales, con distintas frecuencias y amplitudes.

Transformación del espacio de color

RGB-YUV

Convierte la imagen desde el formato de color RGB a YUV, el cual tiene tres componentes:

- Componente Y o iluminación: la imagen en escalas de grises.
- Componente U: diferencia del azul, relativiza la imagen entre azul y rojo.
- Componente V: diferencia del rojo, relativiza la imagen entre verde y rojo.

Las dos últimas son conocidas como crominancia, cuya fase y amplitud corresponden a la saturación y el matiz del color.

RGB-ESCALA DE GRISES

A la escala de grises se le conoce también como grado de claridad que tiene una imagen, por lo que al realizar la transformación lo que se está haciendo es relacionar el valor de este color según la claridad u oscuridad que esté presente. La escala de grises va desde el cero que es el negro hasta el 255 que representa al blanco, por lo que para determinar el valor que le corresponde un color a esta escala se aplica la siguiente ecuación:

$$Escala\ de\ Grises = R \cdot 0.3 + G \cdot 0.59 + B \cdot 0.11$$

Esta es la fórmula que más se aproxima a la detección que realiza el ojo humano, mediante la cual se combinan los tres colores (Scantips.com, 2010) [89]

RGB-HSV (HUE SATURATION BRIGHTNESS)

Mediante este formato se definen los colores en función de sus tres componentes que es el matiz, la saturación y el brillo. Los colores en este formato se representan en forma de cono, ya que de este modo el matiz queda en la región circular, la saturación en función del radio y el brillo en su eje, pudiendo seleccionar cualquier combinación de color en función de estos.

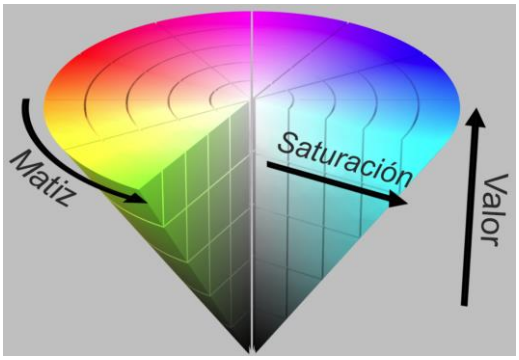


Figura 75: Cono HSV

Para realizar el ejemplo de OpenCV que se muestra más adelante, se utilizó esta gráfica de colores, ya que para trabajar en el formato HSV se ha de seleccionar el rango de los colores que se van a utilizar.

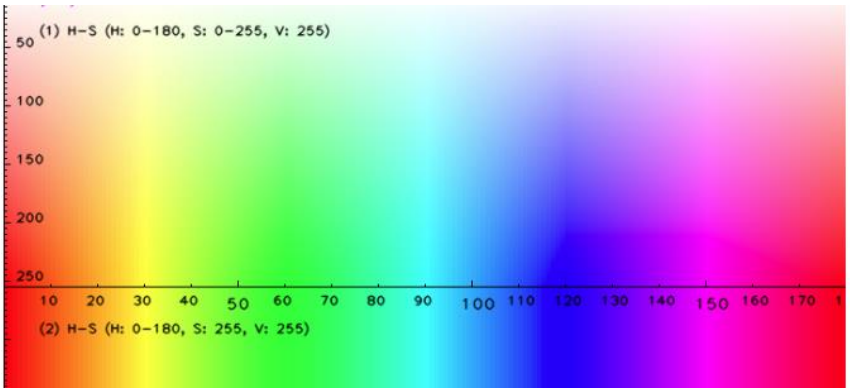


Figura 76: Tabla de colores HSV

- El eje X: representa el matiz del color.
- El eje Y: este representa la saturación.

4.3. Análisis Software

En este bloque se tratarán tres apartados importantes del sistema software, dos de los cuales corresponde al SO Linux y Ubuntu. En este último se mostrarán las mejoras que presenta la versión de la tarjeta Jetson TX2 y se explicará que tres modos de programación se pueden realizar. El tercer apartado y más importante de este bloque trata sobre el kit JetPack 3.1, el cual está formado por un conjunto de herramientas software, APIs (*Application Programming Interface)* [2], módulos y librerías.

4.3.1 LINUX

El sistema operativo de la SBC Jetson TX2 es Linux lo que le permite ser multiplataforma, multiusuario y multitarea. Además, tiene memoria virtual, conexión a red y acceso a códigos abiertos. Este SO se distribuye en diferentes capas en las que se realizan procedimientos concretos y estandarizados, se ha de tener en cuenta que sólo se permite la interacción entre las capas adyacentes. (Documentación informática industrial y comunicaciones,2018) [14]

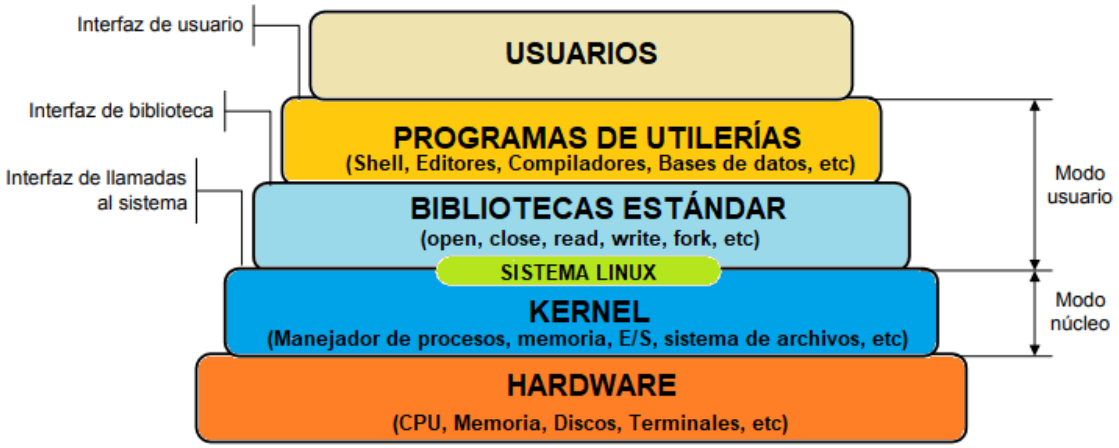


Figura 77: Capas de LINUX

- La capa más inferior corresponde al hardware del sistema, el cual ha sido descrito en el bloque anterior.
- La siguiente capa es el KERNEL o núcleo del SO y está escrito en lenguaje ensamblador. Interacciona con la parte hardware por lo que no es posible su portabilidad entre sistemas.
- La capa superior es la biblioteca estándar y junto con el KERNEL constituyen el SO Linux. Está escrito en el lenguaje C e interacciona con la capa inferior, llamando a sus diversos procedimientos. Esta capa no es dependiente de la máquina y sus procedimientos están estandarizados por lo que es portable.
- La capa consecutiva está formada por los programas de utilerías, su uso hace posible que las tareas del usuario se lleven a cabo, ya que sirve de unión entre las solicitudes del usuario y los procedimientos de la biblioteca.

- La capa más externa es la de usuario y se encarga de traducir el conjunto de información proveniente de las capas inferiores en datos que pueden ser interpretados por el usuario.

LINUX constituye el núcleo del SO y se encarga de realizar las tareas básicas del sistema, dejando el resto de trabajo a las otras capas. Por ejemplo, una de las funciones que realiza es establecer el tiempo de ejecución de los programas. Su lenguaje de programación es de alto nivel como el lenguaje C, pero es compatible además con el lenguaje ensamblador, Perl, Python y con otros lenguajes Shell scripting. La versión que presenta la placa TX2 es la 4.4.38 LTS por lo que da soporte a largo plazo y además cuenta con varias mejoras que hace que el sistema sea más rápido y estable. Alguna de estas mejoras son:

- Tiene mejor arranque del sistema.
- Cuenta con el soporte Open-Channel SSD, que se realiza mediante LightNVM [38]. Esto permite el aislamiento de E/S, una memoria no volátil y una latencia predecible. Todo esto hace posible la conexión interna paralela entre la SSD y el host.
- El soporte de discos independientes RAID 5 presenta mejoras en la recuperación interna de datos. Es más rápido ya que no se realizan los cálculos en el disco donde se ha producido el error de paridad o CRC.
- Presenta varias mejoras en el almacenamiento de archivos.
- El protocolo TCP ha mejorado por lo que la transmisión es segura y estable, ya que se reciben los datos sin errores y en el mismo orden en el que fueron enviados.
- Es compatible con múltiples Drivers.
- Se optimiza el consumo de la memoria RAM.

4.3.2 UBUNTU 16.04 LTS

Se trata de un SO basado en LINUX, financiado y gestionado por Canonical [6]. Su sistema de gestión de paquetes corresponde al SO DEBIAN [15], por lo que el software se mantiene estable y actualizado, además se caracteriza por su rapidez y por ser de código abierto. La programación de este SO la realiza el usuario desde cero, siendo totalmente personalizado, por lo que se requiere de un cierto nivel de programación. Su interacción con la parte hardware es muy buena, ya que detecta con gran facilidad a los periféricos que se le conectan al sistema. Cuando se realiza alguna actualización dependiendo de cuál sea, no es necesario reiniciar el sistema, en otros casos sí. Otra de sus características es que se actualiza continuamente, proporcionando a los usuarios nuevas versiones, pero estas actualizaciones en ocasiones provocan errores en la conexión Wifi, audio o vídeo, por lo que se han de actualizar sus drivers correspondientes. Esta información sobre Ubuntu ha sido tomada de (Lavidaconubuntu.blogspot.com.es, 2018) [48] y (Ventajas y Desventajas De Instalar "UBUNTU", 2011) [106]

Estas actualizaciones pueden ser regulares o LTS, la diferencia entre ellas es que las regulares añaden nuevas funciones poco a poco y suelen ser actualizadas cada nueve meses. Sin embargo, las LTS son estables por lo que las funciones añadidas son mantenidas durante un mayor periodo de tiempo, aproximadamente cinco años. Esta SBC cuenta con la versión 16.04 LTS, por lo que presenta las siguientes características:

- Sistema de paquete Snap: Esto permite que las aplicaciones se actualicen más frecuentemente y sean más unificadas. Además, se mejora en seguridad ya que las aplicaciones están más aisladas del resto del sistema y son más estables. Estos paquetes son compatibles y permiten actualizar sólo la parte que se requiere, por lo que no es necesario actualizar todo el sistema. Por otro lado, permite la opción de instalar nuevas aplicaciones en versiones más antiguas.
- Proporciona un soporte para los archivos Zfs [71].
- Hipervisor LXD [33]: Opera a mayores velocidades y con latencias bajas.
- Es compatible con los servidores Intel y AMD de 32 y 64 bits. También da soporte a mainframes IBM Z, IBM LinuxONE y ARM.
- Proporciona novedades en el entorno del escritorio, ya que se pueden realizar nuevas modificaciones y configuraciones.
- Se ha sustituido el Software Center por GNOME Software, aunque no existe diferencia entre ellas.
- Se incluye la versión 3.5 de Python y otros programas preinstalados que son Firefox 45, Thunderbird 38, Chromium 48, LibreOffice 5.1, Nautilus 3.14.2, Totem 3.18, Rhythmbox 3.3, GNOME Terminal 3.18, Eye of GNOME 3.18 y Shotwell 0.22.

Existe tres modos de instalar este SO, las cuales son las que se describen a continuación:

- Primer modo: Como único sistema del equipo y es el más recomendable, ya que en el disco duro sólo estaría él, evitándose de este modo problemas. Los problemas que se pueden dar si se hace una mala implementación de este es que se pierdan datos o se realice una mala partición. Además, se puede considerar contraproducente ya que se pierde tiempo al cambiar de SO.
- Segundo modo: Consiste en simular su instalación mediante Wubi, es el menos recomendable. Debido a que no se utiliza todo el potencial del sistema, ya que para su funcionamiento depende de Windows.
- Tercer modo: Instalación de ambos sistemas operativos en el mismo equipo mediante dual-boot. Este es el método que se ha utilizado en este caso.

4.3.3 JetPack 3.1

El sistema SBC TX2 se caracteriza por su uso en aplicaciones de inteligencia artificial, ya que cuenta con el kit JetPack desarrollado por NVIDIA. Este SDK (*Software Development Kit*) reúne todo el software de la plataforma Jetson, por lo que está compuesto por un conjunto de herramientas que resultan fáciles de actualizar lo que permite un aprendizaje continuo y de mayor rendimiento (Franklin et al., 2018) [30] y (Anon, 2018) [5].

Linux 4 Tegra R28.1 (L4T)

El procesamiento paralelo de los núcleos permite un mayor rendimiento ya que los algoritmos de imágenes se ejecutan con mayores velocidades, en comparación con el resto de las arquitecturas. Estas últimas procesan los datos en serie, mientras que la arquitectura de NVIDIA

virtualiza la canalización del procesamiento de imágenes del sensor, CPU, ISP y GPU. Además, es capaz de procesar más cantidades de datos en menos tiempo, lo que le permite capturar imágenes con una elevada gama de intensidades de luz, ya que se ha mejorado las técnicas HDR. Las intensidades del medio tienen una relación de 10000:1, en el que cubren desde las zonas más brillantes hasta las más oscuras. Se ha de tener en cuenta que algunos de los factores que determinan el rango de intensidades son el tamaño, la sensibilidad y calidad del sensor CMOS, además del tamaño de cada píxel. Esta arquitectura de NVIDIA utiliza algoritmos más eficaces, ya que combina la potencia masiva de la GPU, CPU e ISP, y los avances tecnológicos de los sensores. Obteniendo mejores resultados de imágenes HDR, aunque en el entorno donde se realice la captura de la imagen exista gran contraste entre zonas con luces y sombras.

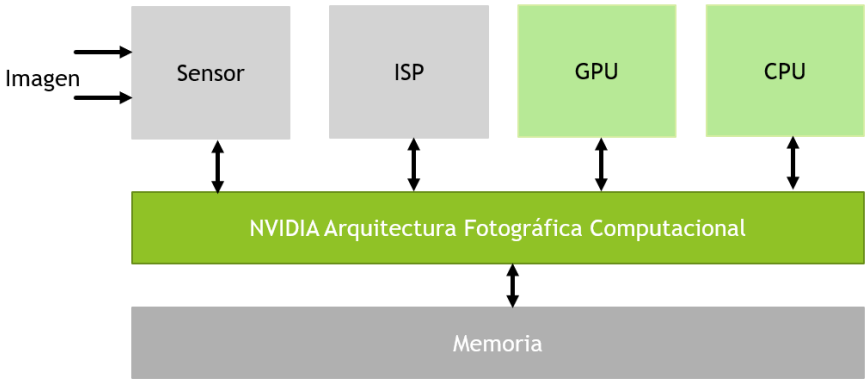


Figura 78: Arquitectura Fotográfica Computacional

Esta arquitectura es compatible con diversos soportes, los más destacados son los siguientes: (NVIDIA Developer, 2018) [64]

- El kernel de Linux de versión 4.4.38, que es la versión Linux de este sistema.
- Espacio de usuario de 64 bits y bibliotecas de ejecución a tiempo real.
- Da soporte a la API Vulkan [65] (NVIDIA Developer, 2018) [68], posibilitando de este modo la ejecución de aplicaciones con gráficos 3D.
- No es necesario que la información sea procesada por el ISP, ya que los sensores de la cámara son compatibles con el controlador multimedia V4L2. Mediante la API V4L2 se posibilita la función de descodificación, codificación, conversión de formato y escalado de video.
- Se puede utilizar la biblioteca C++ de Libargus lo que mejora el renderizado en paralelo. Esta API es síncrona, de bajo nivel, da múltiples soportes y salida de flujo EGL. Si la salida de la cámara CSI es RAW requiere del ISP, por lo que se ha de utilizar el plugin Libargus o GStreamer.
- Es compatible con varias versiones de OpenGL, las más destacadas son las siguientes:
  - OpenGL 4.6 Beta: Es la versión más reciente y realiza un procesamiento geométrico, sombreado y filtrado anisotrópico más eficiente. Además, se obtiene un mayor rendimiento y más información mediante estadísticas y consultas de desbordamiento y contadores.
  - OpenGL 4.5: Una de las características que presenta esta versión es el acceso directo DSA, por lo que se puede modificar objetos OpenGL sin tener que vincularlo al contexto. Esto se debe a que las versiones anteriores a 3.3. utilizan “fixed-pipeline”, en el cual sus funciones son fijas y sólo se pueden cambiar determinados

parámetros como es el color o posición. Mientras que las versiones posteriores utilizan "programable-pipeline", en el cual se puede programar por completo utilizando shaders [50].

- OpenGL ES 3.2: Con esta versión se obtiene mayores detalles, renderizados de procesos en coma flotante y compresión de textura ASTC.
- OpenGL ES 3.1: Esta versión incluye el sombreado independiente de vértices y fragmentos, así como la realización de cálculos de sombreado y la realización de dibujos mediante comandos indirectos.
- EGL 1.5 con EGLImage: Posibilita que diferentes APIs compartan la misma información, esto es posible a EGLImage que trata la información mediante mecanismos específicos para cada API.

- Protocolo de comunicación RandR 1.4, que se encarga de parámetros de la pantalla como su tamaño, rotación y frecuencia de actualización.
- Sistema de gestor de ventanas X11 que se encarga de la interfaz gráfica y de la comunicación con la tarjeta de video. Este procesa las imágenes en base a 8 bits de color, manteniendo la diferencia entre los formatos RGB y YUV.

Cámara V4L2/API de códec

Esta herramienta da soporte a cámaras y dispositivos de vídeos, y además cuenta con la API multiplanar para códecs de vídeos. Los controladores de V4L2 adquieren los datos de un sensor y transfieren esa información al espacio del usuario. La API multiplanar permite utilizar nuevos formatos de píxeles que indican el uso de múltiples planos. Cada memoria puede contener un conjunto de muchos planos, y estos son separados para cada componente de color. Por otro lado, los componentes Y y CbCr deben estar situados en diferentes partes de la memoria. La interfaz de esta herramienta es videobuf2 que está formada por tres tipos de buffers.

- Buffers Vmalloc: Copian los datos y tienen una estructura continua al largo de todo el espacio.
- Buffers DMA contiguos: Están unidos a la memoria, ya que el sistema hardware requiere de un acceso directo a la memoria, y mediante este buffer se puede realizar este acceso.
- Buffers S/G DMA: Estos buffers están colocados a lo largo de la memoria, para que el hardware bifurque o agrupe los datos para el DMA.

Estos buffers cuentan con datos adicionales que permiten realizar diversas funciones como filtros o la detección de contornos.

CUDA Toolkit 8.0.82

Cuenta con un compilador nvcc que se encarga de separar el código del host y del device, siendo este compilador el que posibilita la realización de aplicaciones CUDA en sistemas embebidos y acelera el procesamiento de la GPU (NVIDIA Developer, 2018) [63] (INTRODUCCIÓN A LA PROGRAMACIÓN EN CUDA, 2016) [39]. El conjunto de herramientas que presenta optimiza el rendimiento de las aplicaciones y las más destacadas son:

- Bibliotecas con mayor precisión por lo que se obtiene un mayor rendimiento en tiempo real. Esto posibilita que el procesamiento de imágenes, el aprendizaje profundo y el análisis de gráficos se realicen mucho más rápido. También lleva incorporada una biblioteca con varios ejemplos, cuya ejecución se mostrará más adelante.
- Herramientas utilizadas en la depuración y optimización de imágenes como:
  - Allinea DDT: depura aplicaciones hibridas paralelas en un único núcleo de la GPU. Se utiliza para depurar programas MPI, OpenMp, CUDA y OpenACC.
  - TotalView: está basada en la GUI y permite depurar uno o más procesos.
  - Nsight Nvidia: realiza la depuración y creación de perfiles.
  - CUDA-GDB: permite la depuración simultanea de la CPU y partes de la GPU.
  - CUDA-MEMCHECK: en el acceso a la memoria identifica rápidamente los errores, por lo que se obtiene una mayor optimización. Siendo de este modo más fácil de localizar estos errores y de resolverlos.
- Cuenta con herramientas para realizar el análisis del rendimiento del sistema, algunas de estas son:
  - Visual Profiler Nvidia: esta herramienta posibilita escribir comentarios en los códigos C y C++.
  - TAU Performance System: realiza el análisis de programas híbridos CUDA, pyCUDA y OpenACC, mediante herramientas de rastreo.
  - VampirTrace: con ella se obtiene una visión detallada de los aceleradores durante la ejecución.
  - PAPI CUDA: proporciona el valor del rendimiento de los núcleos GPU.
  - CUPTI: esta se encarga de proporcionar información sobre la CPU.

TensorRT 2.1 GA

Con esta herramienta se obtiene un mayor rendimiento en aplicaciones basadas en Deep Learning, ya que su tiempo de ejecución es de baja latencia. Mediante TensorRT las redes neuronales se separan en capas optimizadas y se implementan en centros de datos a gran escala, obteniendo tiempos de ejecución optimizados. Es posible trabajar con redes neuronales de formato completo (FP32) y reducidos (INT8 y FP16), obteniendo mayores rendimientos con el formato INT8 (NVIDIA Developer, 2018) [66]. La versión de esta tarjeta es la 2.1, lo que significa que trabaja con dos dimensiones de rango uno. Además, cuenta con herramientas como Coffe y TensorFlow, esta última es de código abierto y su implementación se puede llevar a cabo mediante la API de Python.

También permite implementar capas nuevas y únicas, mediante la API de capas personalizadas. Esta API proporciona funciones específicas a los núcleos CUDA, por lo que el usuario puede definir nuevas capas con las características que desee. Todo esto es posible debido a las mejoras que implementa esta herramienta que son:

- Calibración de precisión de peso y activación: Reduce la pérdida de precisión, por lo que aumenta el rendimiento del formato FP32.
- Tensor de capa y fusión: mejora el uso de la GPU y unifica la ejecución del Kernel. Esto se consigue mediante la combinación de nodos sucesivos en un solo nodo, optimizando de este modo el almacenamiento de memoria y el ancho de banda.

- Autoajuste del Kernel: Selecciona la capa más optimizada y los mejores algoritmos paralelos.
- Memoria dinámica del Tensor: Mejora el uso de memoria reduciendo el tiempo de uso, ya que solamente la utiliza durante el tiempo de ejecución dejándola libre el resto del tiempo.
- Ejecución Multi-Stream: Los datos de entrada que recibe los separa en diversos flujos del mismo modelo y peso, posteriormente estos flujos son procesados de forma paralela.

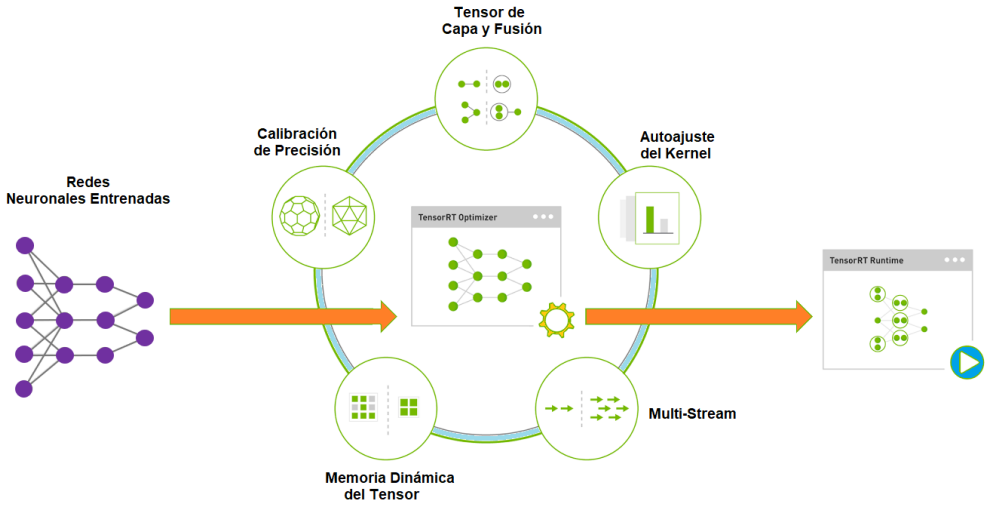


Figura 79: Mejoras deL TensoRT 2.1

VisionWorks 1.6

Es un conjunto de herramientas software que tiene sus propios algoritmos y pipelines para el procesamiento de imágenes y el desarrollo de aplicaciones de visión artificial (NVIDIA Developer, 2018) [67]. Está basado en la API OpenVX y cuenta con un mecanismo que tiene definido el tiempo que dura las imágenes, por lo que cuando el contador de referencia llega a cero, se destruye la imagen mediante Framework VisionWorks. El ciclo de vida de una imagen tiene cuatro fases, que son las siguientes:

- Fase 1: Corresponde a la creación de la imagen y la aplicación obtiene referencias de ella, por lo que el contador toma referencia del tiempo de ciclo correspondiente.
- Fase 2: se lleva a cabo su uso, la aplicación toma su referencia y trabaja con ella. Durante esta fase el contador va disminuyendo.
- Fase 3: en esta fase la aplicación libera su referencia y deja de utilizarla.
- Fase 4: corresponde a su destrucción, ya que se ha dejado de utilizar y el contador está a cero, esta destrucción se realiza mediante Framework VisionWorks.

Cuando se utiliza la API OpenVX se puede dar dos casos, que sea una llamada única o múltiple, en esta última se utilizan múltiples hilos. En el caso de que se trate de una llamada única se puede llamar a cualquier función con cualquier argumento. Sin embargo, si es una llamada múltiple se tienen que cumplir cuatro condiciones, que son:

- Primera condición: Las imágenes deben ser un conjunto de datos o segmentos.

- Segunda condición: Al ser una llamada múltiple se realiza mediante una ejecución secuencial, por lo que estás imágenes no pueden ser destruidas hasta que no finalice la ejecución por completo.
- Tercera condición: Si en las diferentes llamadas se utiliza el mismo búfer como argumento, este no se ha de modificar ya que perturbaría al resto de las llamadas.
- Cuarta condición: Si una de las llamadas está trabajando con un gráfico el resto de las llamadas no pueden trabajar con parámetros de este gráfico, excepto si se el dato con el que se quiere trabajar trata de un dato de entrada para todas las llamadas.

En la siguiente tabla se muestran algunos de los ejemplos que incluye esta herramienta software. A un lado se muestran ejemplos básicos y en el otro, ejemplos que requieren de pipelines para el procesamiento de datos.

BÁSICOS	PIPELINES
Operaciones aritméticas	Seguimiento de funciones
Transformaciones geométricas	Captura de imágenes
Captación de características	Interoperación Open-CV-NPP-OpenVX
Filtros y Análisis de imágenes	Extracción de profundidad estéreo
Flujo y profundidad	Detección de círculos y líneas

Tabla 19: Ejemplos de operaciones

Tegra System Profiler 3.8 (TSP)

Esta herramienta se encarga de proporcionar información sobre el comportamiento de la CPU. Su funcionamiento se basa en realizar perfilados, muestreos y seguimientos, obteniendo de este modo los perfiles estadísticos del comportamiento del sistema.

- El perfilado consiste en recopilar los valores de rendimiento, por lo que se realiza un muestreo y rastreo de datos.
- Mediante el muestreo se detiene periódicamente el proceso y recopila las trazas de los hilos activos. Obteniendo de este modo el tiempo que se tarda en ejecutar cada función. Este resulta muy útil para detectar los cuellos de botella que se dan en la CPU.
- Seguimiento de los procesos que se están llevando a cabo mediante el TSP.

Este muestreo se lleva a cabo mediante diversos algoritmos como por ejemplo el retroceso de arriba- abajo, proporcionando información de todo el sistema incluyendo el SoC, CPU y GPU. Cuenta además con contadores de ARM PMU, lo que le permite contar ciclos, detectar fallos y realizar interrupciones.

API Tegra Multimedia 27.1

Este conjunto de herramientas facilita el desarrollo de aplicaciones flexibles, debido a que es un conjunto de API de bajo nivel. Al estar situado en niveles inferiores realiza una mejor interacción con la parte hardware del sistema mejorando de este modo el control de esta parte. Dos de

estas herramientas de las cuales ya se han hablado anteriormente son Libargus y V4L2, que realizan codificación, descodificación, escalado y otras operaciones con imágenes. La herramienta NVOSD se utiliza para la visualización de las imágenes en la pantalla y mediante Buffer Utility, se puede utilizar un búfer de modo compartido.

CuDNN 6.0 (Cuda Deep Neural Network)

Es una biblioteca que acelera los procesos de entrenamiento de redes neuronales de aprendizaje profundo realizados por la GPU (NVIDIA Developer, 2018) [65]. Se utiliza especialmente en aplicaciones basadas en redes neuronales convolucionales tanto de tipo feedfordward como feedback, y admite los formatos FP32, FP16 y INT8. Los algoritmos utilizados permiten una fácil implementación de redes neuronales con diferentes dimensiones y parámetros. Es compatible con varios entornos como TensorFlow, Caffe2 o MATLAB.

CuDNN permite realizar e implementar diseños personalizables para el entrenamiento de las redes neuronales. En los que valores de parámetros como la dimensión o stride de los tensores 4D son flexibles, lo que le permite ser de fácil integración en cualquier red neuronal. Además, esto le evita que sea necesaria su transformación a la entrada o salida de la red, como sucede a veces en otros algoritmos como GEMM (*General Matrix Multiply*), por lo que en general el rendimiento que ofrece CuDNN es muy alto y a diferencia de otros algoritmos utiliza menos memoria. (Docs.nvidia.com, n.d.) [13]

GStreamer 1.8.2

Es una plataforma de código abierto y baja latencia, con la cual se puede realizar múltiples aplicaciones de audios y videos. Su lenguaje de programación es C, aunque es compatible con otros lenguajes, y su funcionamiento consiste en codificar o descodificar flujos de datos digitales. Presenta una estructura similar a una tubería, por lo que se le denomina *pipeline,* en cuyo interior se encuentran los elementos que comunican a los distintos dispositivos hardware. Estos últimos son los encargados de realizar la decodificación y codificación de las señales, mientras que los elementos son los encargados de transferir estas señales. La transferencia de estos datos es rápida y ligera, aumentando de este modo el rendimiento del sistema. La versión 1.8.2 es similar a la anterior con la diferencia de que esta incluye la corrección de errores que había en versiones anteriores (Gstreamer.freedesktop.org, 2018) [32]

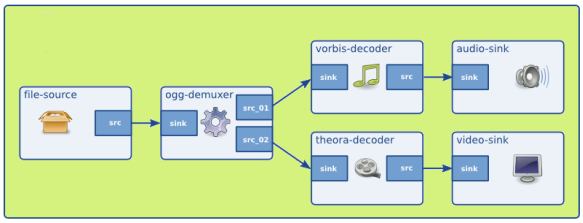


Figura 80: Flujo de transferencia de datos

Su estructura interna está formada por cuatro componentes:

- Elemento: es el medio por el cual se transfieren los datos. Cada uno realiza una función específica, por lo que, colocándolos en cadenas, los datos son procesados obteniendo en la salida el resultado final de los datos tratados.
- Bins: es un conjunto de elementos que puede ser tratados como un elemento independiente. Por lo que si se cambia el estado de este elemento automáticamente se está cambiando el estado de sus elementos internos.

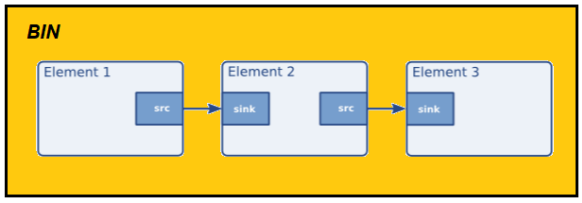


Figura 81: Estructura BIN

- Pipeline: este componente está formado por un grupo de bins y determina la sincronización de los bins internos. Al ejecutarlo se inicia el procesamiento de datos de modo secuencial y este flujo de información se transfiere hasta llegar al final del pipeline.

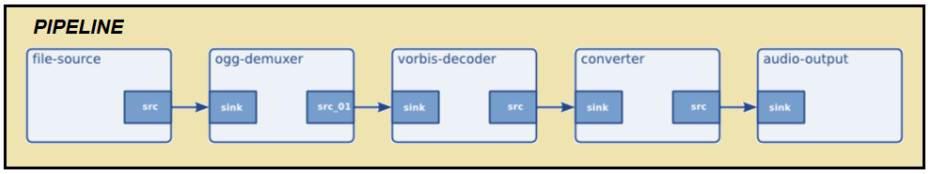


Figura 82: Estructura PIPELINE

- Pads: son las entradas y salidas de los elementos, se utilizan para permitir o denegar la transmisión de flujos de datos entre distintos elementos. Para que sea posible la transferencia los datos han de ser compatibles, consiguiendo de este modo que el conjunto de datos viaje desde el elemento fuente hasta los receptores.

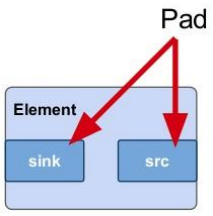


Figura 83: Pad

Existen diferentes modos para llevar a cabo la transferencia de datos, esto se puede realizar mediante buffers, eventos, mensajes y consultas (Genie.webierta.skn1.com, 2018) [31]. Los buffers son una especie de canales unidireccionales que transfiere los datos entre los elementos.

Los eventos son similares a los buffers, pero en este caso son bidireccionales y permiten la interacción entre la aplicación y el elemento. Los mensajes son producidos por los elementos y enviados al bus y a la aplicación, estos se utilizan para transmitir información de manera segura y puede ser síncrono o asíncrono. Sin embargo, la aplicación lo gestiona de manera asíncrona mediante su hilo principal. El último modo es realizar la transferencia mediante consultas, que son enviadas por la aplicación para solicitar información sobre el proceso. Las respuestas generadas son síncronas y además funciona como los eventos, ya que es bidireccional.

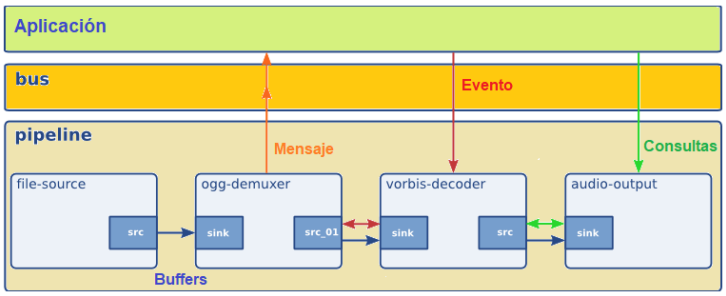


Figura 84: Modos de transferencia de datos

OpenCV4Tegra 2.4.13

Es una biblioteca desarrollada por Intel de código abierto y multiplataforma, ya que es compatible con diversos SO (Acodigo.blogspot.com,2018) [2]. Su lenguaje de programación es C, aunque es compatible con otros como Phyton, MATLAB y Ruby. Tiene más de 2500 funciones, lo que le hace una aplicación de fácil, rápida y eficiente ejecución para el desarrollo de aplicaciones de visión artificial. También cuenta con las bibliotecas IPP y MLL, siendo la primera utilizada para realizar aplicaciones de tiempo real, mientras que la segunda posibilita la visión por computadora y el auto aprendizaje. Esta última biblioteca se basa en algoritmos de reconocimiento estadístico y agrupación de patrones.

La librería OpenCV cuenta con diversas funciones con las que se pueden realizar múltiples aplicaciones como por ejemplo la detección de objetos, la restauración de imágenes, la extracción del esqueleto de un objeto o la de los canales RGB.

La versión de esta tarjeta Jetson TX2 es la 2.4.13 que es de código cerrado, pero de uso gratuito. A diferencia de las de código abierto, esta presenta un conjunto de optimizaciones vinculadas a la CPU multinúcleo, a la GPU GLSL y a la ARM NEON. Esta biblioteca se instala automáticamente cuando se implementa el JetPack de NVIDIA y en el caso de que se trabaje con grandes imágenes es mejor utilizar la GPU, mientras que si se trata de imágenes pequeñas es mejor la CPU. Para realizar las pruebas y ejecutar las aplicaciones se decidió instalar la versión de OpenCV 3.4.1. Una vez instalada se observó que esta versión no era compatible con el programa YOLO, por lo que se tuvo que desinstalar e instalar la versión 3.4.0. Esto será explicado con mayor detalle más adelante.

GPU GLSL

La mejora que presenta se debe al shader de fragmentos y vértices, que permite el procesamiento de gráficos sin la necesidad de un lenguaje específico. Se caracteriza por ser compatible con diversos SO y por ser multiplataforma, ya que es compatible con distintas tarjetas gráficas de lenguaje OpenGL (Acodigo.blogspot.com,2018) [1]. Los shaders son pequeños programas ejecutados por la GPU, que se transfieren desde la CPU hasta la pantalla, mostrando de este modo la imagen procesada. Para conseguir esto las funciones son transferidas por el pipeline de OpenGL, el cual está formado por un conjunto de etapas programables y no programables. Depende de que etapa sea, se podrá acceder a la modificación de determinados parámetros siempre que esta sea programable. Un ejemplo de etapa programables es Vertex Shader que se utiliza para procesar la información geométrica de una figura 3D compuesta por miles de vértices. Cada vértice contiene una determinada información por lo que Vertex Shader se ejecutará tantas veces como vértices tenga, ya que cada vez que se ejecuta accede a un vértice diferente. Otro ejemplo de etapa programable es Fragment Shader cuyo funcionamiento es procesar cada uno de los fragmentos obtenidos de etapas anteriores. Cada vez que se ejecuta procesa uno de estos fragmentos, que son un conjunto de datos que contienen información sobre el píxel.

Lo más común en Vertex Shader es realizar escalado o rotación a las imágenes, mientras que en Fragment Shader se determina la textura, color o iluminación de la imagen.

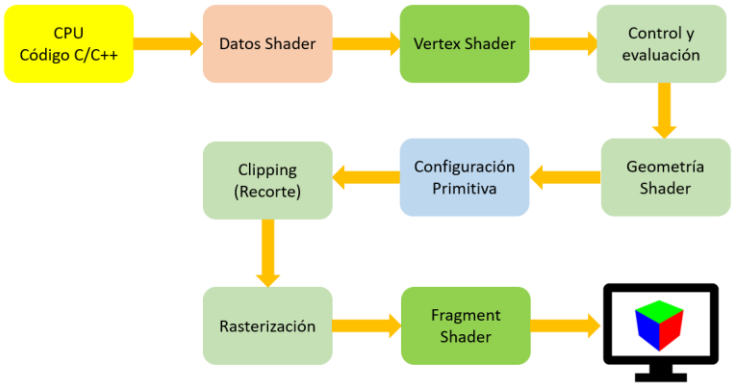


Figura 85: Pipeline OpenGL

ARM NEON SIMD

Con las optimizaciones de la tecnología NEON es posible acelerar los algoritmos y funciones de codificación y descodificación de audios y videos. Esta tecnología corresponde a la arquitectura SIMD, en las que las instrucciones realizan las mismas operaciones en todos los vectores, ya que sus registros son considerados como vectores del mismo tipo de datos.

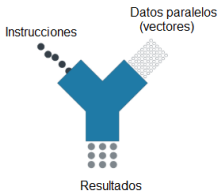


Figura 86: Arquitectura SMID

Esta SBC TX2 es compatible con las versiones ARMv8 de 32 y 64 bits, y también con la versión ARMv7, por lo que sus tipos de datos son los siguientes según la arquitectura:

	ARMv7	**ARMv8**	
		AArch32	AArch64
Coma flotante	32 bits	16 y 32 bits	16, 32 y 64 bits
Entero	8, 16 y 32 bits	8, 16, 32 y 64 bits	8, 16, 32 y 64 bits

Tabla 20: Tipos de datos

Capítulo 5

Experimentación con ejemplos y demos incluidas

Lo primero que se descargó e instaló fue el JetPack 3.1, para lo cual se siguió el tutorial de (JetPack 3.0 - NVIDIA Jetson TX2. (2017). [video]) [103]. Posteriormente para comprobar su correcta instalación se ejecutaron varios ejemplos que se muestra en el siguiente enlace (Testing CUDA Programs on NVIDIA JETSON TX2, 2018) [105]. Después las demos que se muestran a continuación fueron descargadas de la plataforma Github e implementadas posteriormente en la tarjeta Jetson TX2. Se ejecutaron y se realizaron varias pruebas, comprobando de este modo su eficiencia y rendimiento mediante los resultados obtenidos. Durante la realización de las pruebas se tuvo un pequeño inconveniente, y fue que los objetos utilizados para realizar las pruebas fueron aquellos que se tenía a mano dentro de casa. Debido a que no fue posible trasladar al exterior la tarjeta Jetson TX2 ya que requiere de una pantalla de monitor para observar los resultados y del teclado y ratón para interactuar con el sistema. Como solución alternativa se utilizó una televisión en la que se proyectaba un vídeo y la cámara incorporada en la tarjeta enfocaba y capturaba las imágenes de esta. Con los otros modos para le ejecución de las pruebas no se tuvo ningún problema, ya que las imágenes y videos fueron descargados de internet.

Al final del vídeo que se siguió para la instalación del JetPack 3.1 se muestra la ejecución de un ejemplo de detección de vehículos. Este consiste en ejecutar un programa en el que se visualiza un vídeo y señala con un cuadrado rojo los vehículos que son detectados. Pero al ejecutarlo se obtuvo un problema, ya que el archivo con el comando para su ejecución no se encontraba en la carpeta correspondiente. Al buscar una posible solución se observó en algunos foros que más personas habían tenido el mismo problema y daban como solución el conjunto de líneas que deberían estar en la carpeta. Por lo que se ejecutaron, pero se comprobó que con esas líneas ya no funcionaba. Se continuó buscando y se encontró en (Devtalk.nvidia.com,2017) [11] las líneas correctas con las que se pudo llevar a cabo la ejecución del ejemplo.

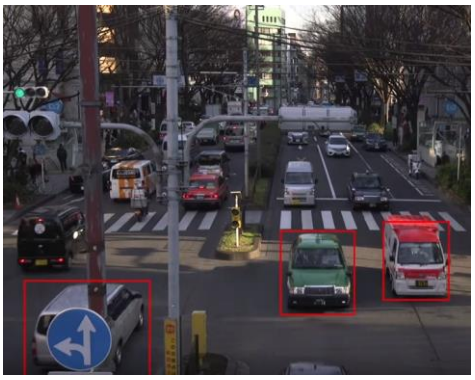


Figura 87: Detección de vehículos

5.1.Ejemplos de CUDA 8.0

Al instalar el Jetpack 3.1 el módulo CUDA 8.0 viene incorporado y este contiene algunos ejemplos que se ejecutaron para observar en que consistían e iniciar así las comprobaciones del funcionamiento del sistema.

5.1.1 Partículas en movimiento

Este ejemplo consiste en la iteración de partículas 3D, el cual se suele utilizar en simulaciones de la dinámica de los fluidos. La representación de las partículas se realiza en base a OpenGL, que mediante el sombreador de GLSL, hace que los puntos representados en el espacio tengan apariencia esférica. Para la representación de este código lo que se realiza es la división del espacio de simulación en cuadrículas pequeñas, en la que en cada una será representada una partícula. Estas tienen tres parámetros que las identifican que son la velocidad, posición y fuerza, y hay dos más que son la gravedad y la amortiguación. Por lo que la velocidad a la que se desplazan las partículas dependerá de la gravedad y de si está colisionan con otras de su entorno o chocan con el cuadro delimitador. Que a su vez esta velocidad actualizará la posición de la partícula, provocando así el efecto de desplazamiento de estas.

Al ejecutar el ejemplo aparece una esfera roja y al lado un cubo delimitador en cuyo interior se encuentra un conjunto de partículas multicolor formando un cubo pequeño, el cual se descompone y sus partículas se desplazan hacia bajo.

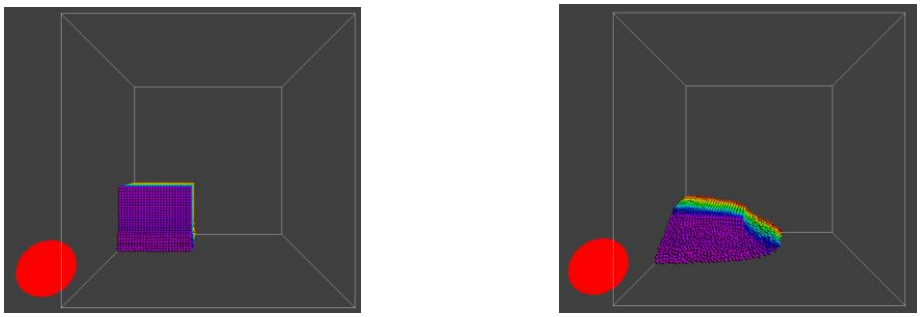


Figura 88: Partículas-Cubo

Si no se ejecuta ningún comando el cubo empieza a girar y el conjunto de partículas forman una esfera, la cual se desplaza hacia abajo y se descompone.

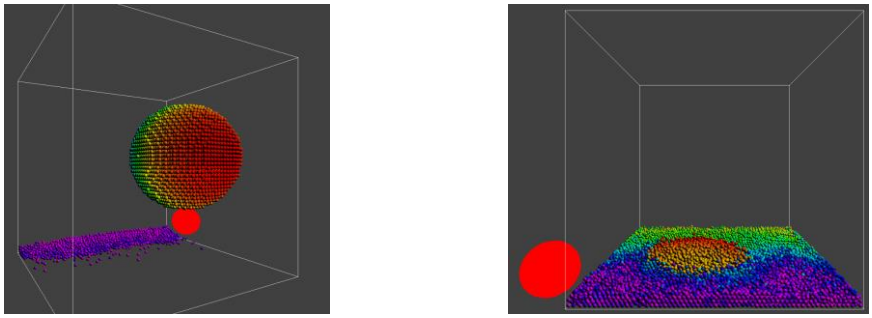


Figura 89: Partículas-Esfera

Al hacer click con el botón derecho aparecen las siguientes opciones.

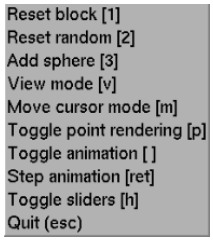


Figura 90: Partículas Opciones

Con “Reset block” las partículas forman nuevamente el cubo, con “Reset Random” las partículas se descomponen al azar y aparece lo siguiente.

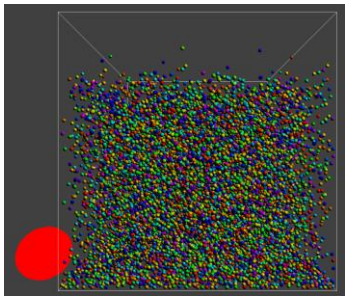


Figura 91: Partículas-Random

Con la tercera opción “Add sphere” lo que se añade es una esfera con nuevas partículas la cual se desplaza hacia abajo y se descompone.

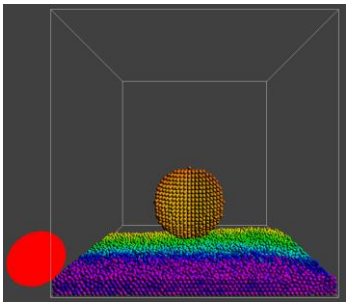


Figura 92: Partículas- Esfera Nueva

La opción “View Mode” permite girar el cubo modificando así el modo de vista, mientras que “Move cursor mode” permite desplazar y situar el círculo rojo donde se desee. Por ejemplo, si se coloca la esfera roja en el interior del cubo, esta interacciona con las partículas como se observa en la siguiente imagen.

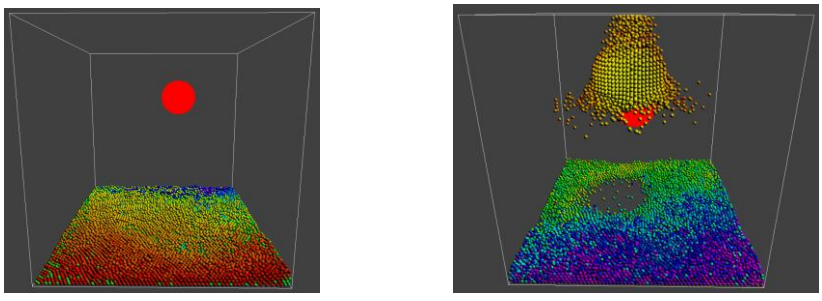


Figura 93: Esfera roja

Con “Toggle point rendering” se cambia la presentación de las partículas, en las siguientes imágenes se muestran dos modos distintos de su visualización.

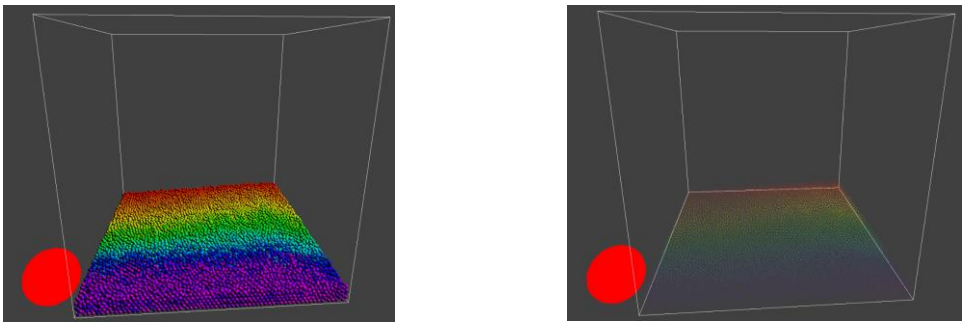


Figura 94: Presentación de Partículas

Por último, con “Toggle sliders” se pueden cambiar los valores de algunos parámetros como el radio de la esfera. En la primera imagen se muestra una esfera de valor 20 de radio, mientras que la segunda es de 3.

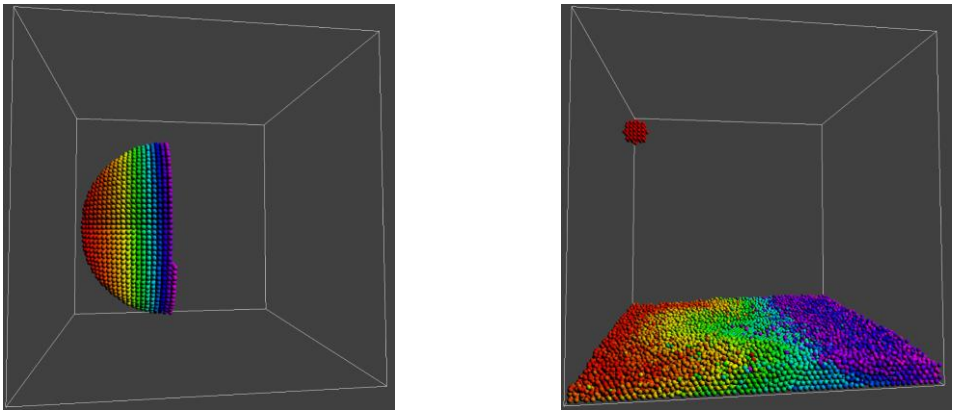


Figura 95: Esferas de distintos radios

5.1.2 Fluido

El siguiente ejemplo simula el movimiento y la viscosidad de un fluido, en el que aparece una ventana con puntos colocados aleatoriamente. La representación de estos puntos se realiza con OpenGL y estas se desplazan con el tiempo, ya que su posición depende del vector de velocidad. Su funcionamiento consiste en convertir la fuerza que se ejerce con el ratón en un vector de dos dimensiones, el cual se relaciona con la velocidad de los puntos. El punto donde se ejerce la presión con el ratón se toma como punto central y se calcula el valor de la fuerza en función de la distancia que se encuentre con respecto a este. Por lo que los puntos que se encuentran más lejanos tendrán una fuerza menor y por tanto menor velocidad. Para darle el aspecto de viscosidad se utiliza el valor de la velocidad, ya que a medida que avanza el tiempo el valor de ese punto será sustituido por el anterior.

Cuando se procesa este ejemplo aparece una ventana de color verde, al interactuar con el ratón este se comporta como un fluido viscoso. Generando una especie de movimiento del fluido por donde se desplaza el ratón.

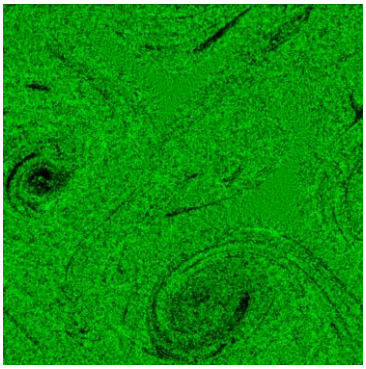


Figura 96: Fluido

5.1.3 Random Fog

Este ejemplo genera un conjunto de puntos que crean una determinada forma con 200000 coordenadas aleatorias. La representación de estos puntos se realiza mediante OpenGL, el cual dependiendo de lo indicado por el comando los distribuirá de una forma u otra.

Al ejecutar el programa aparece una esfera no definida, pero esta forma se puede cambiar y generar un cubo, un plano y una esfera con el contorno marcado.

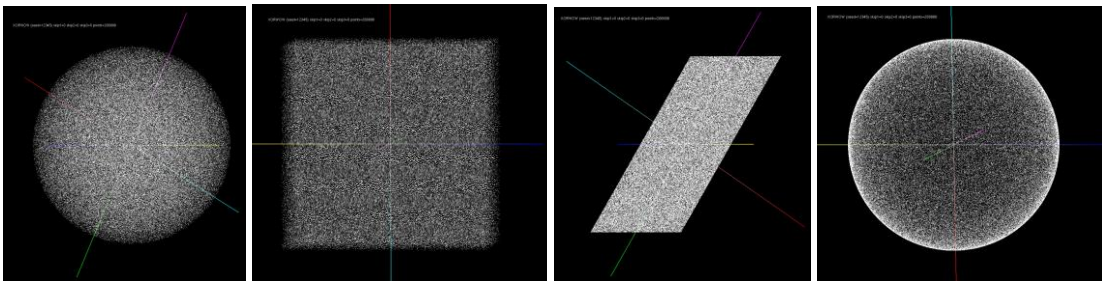


Figura 97: Diferentes formas

En la siguiente tabla se muestran las funcionalidades de algunos de los comandos, en el caso de cambiar la forma geométrica da igual si se selecciona mediante la letra mayúscula o minúscula, ya que funciona con los dos modos.

Comando	Función
Tecla “S”	Genera la forma de una esfera no definida
Tecla “B”	Genera la forma de un cubo
Tecla “P”	Genara un plano
Tecla “E”	Genera una esfera con el contorno marcado
Tecla “+”	Aumenta el número de puntos (Máximo 200000)
Tecla “-”	Disminuye el número de puntos (Mínimo 8000)

Tabla 21: Comandos de RandomFog

Las siguientes imágenes muestran una esfera con tres ejemplos de números de puntos. La primera es de 200000, la siguiente es de 11200 y la última está formada con 8000.

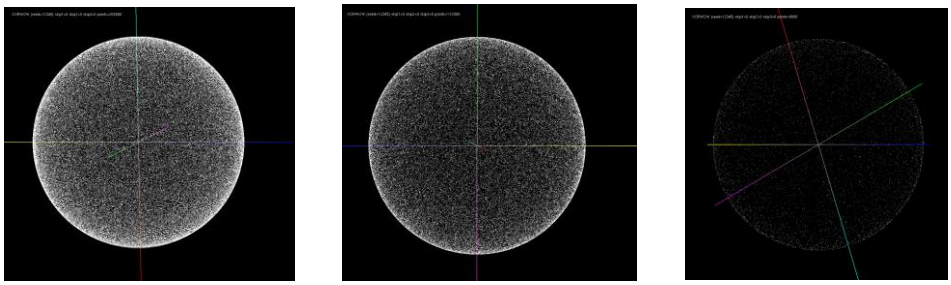


Figura 98: Cambio del número de puntos

En la última imagen superior se observa con más claridad los tres ejes (X, Y y Z) los cuales son representados con distintos colores.

- Eje X positivo: Color amarillo.
- Eje X negativo: Color azul.
- Eje Y positivo: Color magenta.
- Eje Y negativo: Color verde.
- Eje Z positivo: Color cian.
- Eje Z negativo: Color rojo.

5.2. Jetson-Inference

Las pruebas que se muestran a continuación se desarrollaron con el módulo Jetson-Interference, que se basa en el aprendizaje de la red neuronal, y se apoya en la herramienta TensorRT. Al descargar esta demo vienen incorporadas las redes neuronales alexnet y googlenet, las cuales se pueden ejecutar de tres modos, para llevar a cabo estas pruebas se tomó como referencia (GitHub, n. d.) [33] [109].

- ImageNet: Reconoce los objetos de una imagen.
- DetectNet: Detecta la presencia de un objeto en una imagen.
- SegNet: Realiza una segmentación de todo el conjunto de la imagen.

Su ejecución se puede llevar a cabo mediante dos modos, una es de manera consola, en el que las entradas son fotografías y el otro modo es mediante la cámara, detectando a tiempo real su entorno. Se ha de tener en cuenta que se obtienen mejores resultados cuanto más clara y de mejor cálida es la imagen. En el caso de identificación de objetos, su funcionamiento es mejor cuantas menos cosas hay en la imagen.

5.2.1 ImageNet

Su funcionamiento consiste en recibir una imagen como entrada y analizar cada uno de sus píxeles, los cuales son etiquetados mediante valores binarias. Dependiendo de este valor son clasificados como pertenecientes a una categoría de los datos entrenados. Como salida genera

una imagen donde se muestra el porcentaje de probabilidad que tiene con respecto al ejemplo que más se le parece de los clasificados.

Esta red ha sido entrenada con la base de datos ILSVR12 de Imaginet la cual incorpora un millón de imágenes clasificadas en mil ejemplos.

Se realizaron varios ejemplos, en los que se obtuvieron diferentes porcentajes de parecido con los ejemplos de las librerías, algunos de ellos son los que se muestran a continuación.

ImageNet-console

Mayor porcentaje de acierto

En ambos casos se obtuvieron un porcentaje superior al 98 % de probabilidad de parecido con algún ejemplo de la librería.

Figura 99: ImageNet-Console

Menor porcentaje de acierto

En este caso los resultados obtenidos no fueron muy buenos. Por un lado la primera imagen fue identificada de manera errónea, ya que se le reconoció como alas, siendo un avión. Además el porcenataje de su parecido fue del 55.6 %. Esto se debe a que la imagen no es muy precisa, en el que se puede llevar a la confusión de que son unas alas abiertas de un ave. La segunda imagen corresponde a una manzana roja, en este caso si que tiene un porcentaje de parecido alto, pero la identificación es errónea, ya que la identifica con un 86.5 % de parecido con una granada.

Figura 100: ImageNet-Console

ImageNet-camera

Estas pruebas se realizaron con objetos cotidianos y para obtener mejores resultados, se colocó un fondo negro para facilitar así su reconocimiento. Pero al ejecutar el programa se obtuvo el problema de que la imagen generada por la cámara estaba girada 180°. El modo de como corregir la posición de la imagen se obtuvo en (Devtalk.nvidia.com,2017) [12]. En él se explica que una línea del archivo gstreamer ha de ser modificada, cambiando el parámetro "flip-method=2" por "flip-method=0", una vez cambiada la línea se ha de compilar nuevamente el archivo.

Mayor porcentaje de acierto

En los dos casos se obtuvieron muy buenos resultados, con un 100 % de parecido a un oso de peluche en el primer caso y con un 98.14 % de parecido a una plancha.

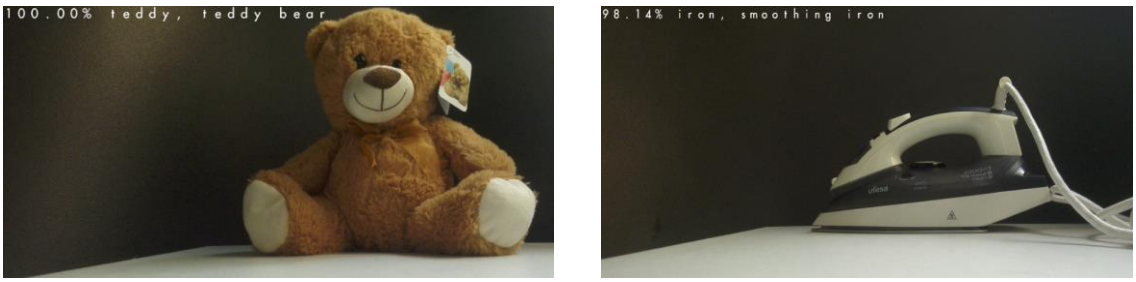


Figura 101: ImageNet-Camera

Menor porcentaje de acierto

En los dos casos siguientes los resultados obtenidos no superaron el 45 %, debido a que en la librería no se encuentra un ejemplo exacto de estos objetos. Estos fueron identificados con objetos con algún rasgo parecido. Para el caso de la tijera, al tener el mango de color negro, la imagen se tomó sobre un fondo blanco. Esta fue identificada con un 40.36 % de parecido con un abrecartas o cortador de papel. Mientras que la calculadora fue reconocida como un control remoto con un 44.26 %.

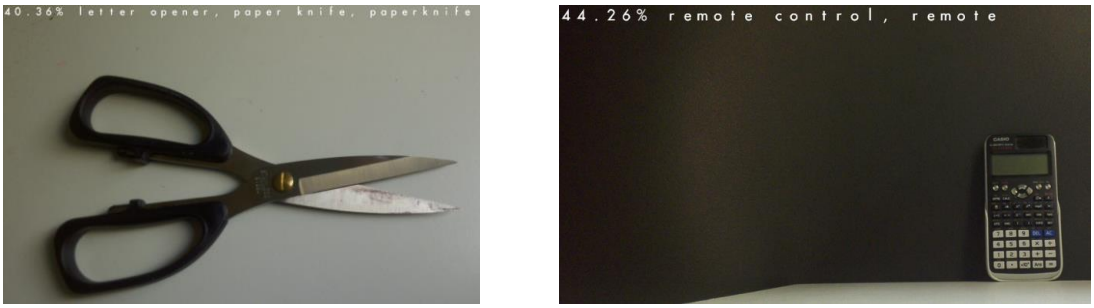


Figura 102: ImageNet-Camera

5.2.2 DetectNet

A diferencia del anterior, este indica si el objeto que aparece pertenece a una clase específica y su localización en la imagen. En el caso de que sea detectado un objeto su ubicación se representa mediante un cuadro delimitador. Esto se obtiene utilizando la red detectNet la cual incorpora una librería compuesta por modelos de reconocimiento pretratados. La base de datos que utiliza en este caso es COCO (*Common Objects in Context*) a continuación, se muestran algunas características de COCO.

- Realiza la segmentación de objetos.
- En imágenes super pixeladas es capaz de realizar la segmentación.
- Su base de datos contiene 330000 imágenes, de las cuales más de 200000 están etiquetadas.
- Detecta 80 categorías de objetos.
- Al menos hay 5 cosas detectadas por imagen.

Sin embargo, la detección de rostros la realiza con la base de datos FDDB (*Face Detection Data set and Benchmark*) que está formada por 2845 imágenes en las que aparecen 5171 caras. (FDDB: A Benchmark for Face Detection in Unconstrained Settings, n.d.) [28]

Los modelos pretratados que incluye la librería son:

Modelo	Lo que detecta
DetectNet-COCO-Airplane	Aviones
DetectNet-COCO-Bottle	Botellas
DetectNet-COCO-Chair	Sillas
DetectNet-COCO-Dog	Perros
Ped-100	Peatones
Multiped-500	Peatones y equipaje
Facenet-120	Rostros

Tabla 22: Modelos de detectNet

DetectNet-console

Mayor porcentaje de acierto

En uno de los ejemplos realizados se utilizó el detector de caras obteniendo un 100 % de aciertos. En el segundo caso se utilizó el modelo de detección de botellas en el que se obtuvo como resultado un 87.5 % de acierto.

Figura 103: DetectNet-Console

Menor porcentaje de acierto

En los siguientes casos se obtuvieron valores igualmente altos, pero a diferencia de los anteriores los valores son inferiores. En la primera imagen se utilizó el modelo pretratado de detección de aviones en el que se obtuvo un 75% de acierto. Mientras que en el segundo caso se realizó la detección de peatones en el que se obtuvo un 85.7 %, este dato corresponde a que se detectó a 12 de las 14 personas que aparecen en la imagen. En este último ejemplo se ha de tener en cuenta que la dificultad de detección de las personas es superior al de las botellas, por poner un ejemplo.

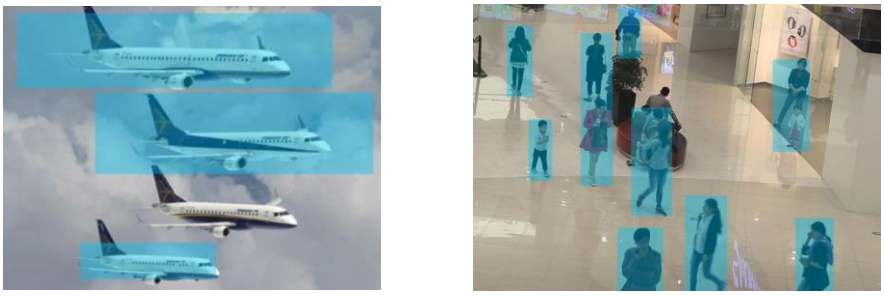


Figura 104: DetectNet-Console

DetectNet-Camera

Mayor porcentaje de acierto

El siguiente ejemplo muestra que la red detectNet incluso utilizando el modo de cámara es capaz de detectar objetos en diferentes posiciones. Para ello se colocaron botellas de aguas en dos posiciones diferentes, como se muestra a continuación. En este caso se utilizó el modelo pretratado de detección de botellas.

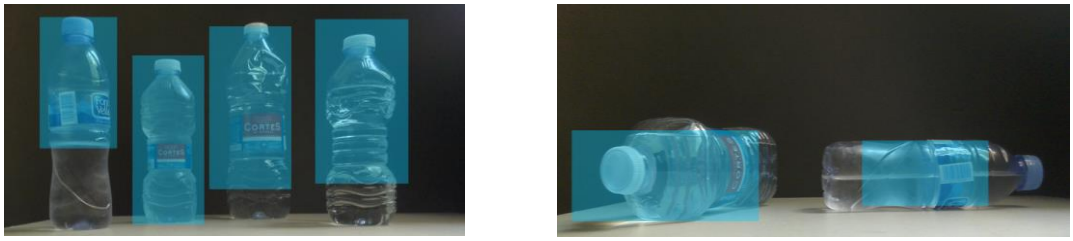


Figura 105: DetectNet-Camera Diferentes posiciones

Como ya se ha explicado anteriormente se utilizó una televisión en la que se reprodujeron los videos para que la cámara lo detectase. Este método fue utilizado para realizar los siguientes ejemplos, en los que se muestran a continuación se obtuvo un 100 % de efectividad, y en él se utilizó el modelo de detección de perros en el primer caso y el de peatones en el segundo.

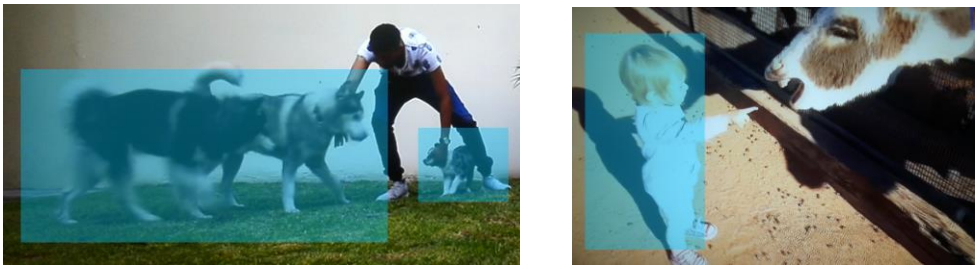


Figura 106: DetecNet-Camera

Menor porcentaje de acierto

En los siguientes ejemplos no se obtuvieron resultados tan buenos. En el primer caso se utilizó el modelo de detección de peatones y como resultado se detectó a la persona, pero además a una parte del perro. En el siguiente caso se utilizó nuevamente el modelo de perros y erróneamente detecto que el leopardo era similar a un perro.

Figura 107: DetectNet-Camera

5.2.3 SegNet

En este caso se cuenta con dos bases de datos distintas, una de ellas posee 21 clases de paisaje y la otra realiza la segmentación en base a 20 clases, esta última corresponde a Pascal VOC y la lista de los 20 objetos se muestra en anexos.

En el primer modo cada píxel es etiquetado según los valores de fondos con los que hubiera sido entrenada la red, por lo que se consigue delimitar el paisaje y señalar aquellas zonas donde se produce un fuerte cambio de color o iluminación.

En el otro modo cada píxel es etiquetado con un número determinado que lo relaciona a una clase de objeto, o en el caso de que no pertenezca a ninguna de estas clases lo etiqueta como fondo de la imagen.

SegNet-Console

Realizando las pruebas se observó que los resultados que se obtuvieron no eran muy buenos, esto se puede deber a que resulta difícil que la red segmente bien una imagen en base a datos de paisajes. A continuación, se muestran algunos de los ejemplos realizados, en el lado derecho se muestra la imagen original y a la izquierda la segmentada.

Figura 108: SegNet-Console playa

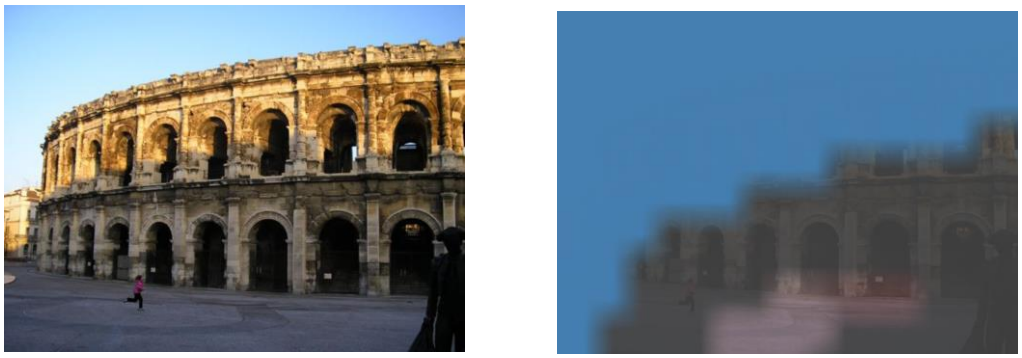


Figura 109: SegNet-Console

Como ya se ha dicho anteriormente la red neuronal de este algoritmo se puede reentrenar, esto se realiza mediante la aplicación web DIGITS (GitHub, n. d.) [34]. Esta plataforma realiza entrenamientos profundos en base a imágenes y es compatible con Caffe, Torch y TensorFlow.

Aunque el entrenamiento de la red neuronal que se presenta más adelante no se realizó mediante DIGITS, se explicará brevemente el funcionamiento de esta.

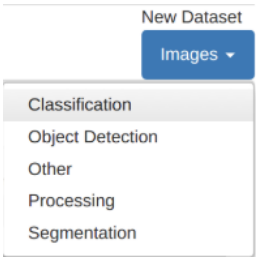


Figura 110: Opciones de DIGITS

Por ejemplo, ImageNet puede ser entrenado de dos modos, en uno de ellos se utiliza como base el conjunto de imágenes ya clasificadas, mientras que el otro modo se inicia desde cero. Mediante el primer modo se realiza una nueva clasificación de las clases de objetos, en la que no se distinguirán los distintos tipos de objetos como razas de perros o marcas de coche. Para ello se descarga el millón de imágenes que ocupa 100 GB de memoria, por lo que es necesario un disco duro externo para su almacenaje. Este modo no permite clasificar nuevos objetos, mientras que el segundo modo si, ya que este se inicia desde cero con imágenes propias. En ambos casos se utiliza la opción “Classification” de DIGITS, mientras que para el entrenamiento de la detección de nuevos objetos se utiliza “Object Detection”.

5.3. Jetson-reinforcement

Está basado en un aprendizaje de refuerzo profundo en el que las redes neuronales convolucionales aprenden en base al entorno de sus agentes y tiene en cuenta la diferencia entre los datos actuales y anteriores. Este módulo lleva incorporada un conjunto de librerías con las que se realizan las simulaciones tanto en 2D y 3D. Este conjunto de tutoriales se basa en un aprendizaje en el que un objeto es capaz de identificar visualmente su entorno y desplazarse hacía objetos de interés desde cualquier posición inicial. Para ello el agente observa el estado de su entorno y elige realizar una determinada acción, la cual es devuelta en forma de dato que indica los efectos que ha tenido. En la siguiente simulación el entorno cambia por lo que ha de realizar nuevamente el análisis de su entorno y optimizar el modelo. Estos módulos de trabajo y este modo de trabajar son muy utilizados ya que una vez entrenados son implementados en robots reales. Se realizaron tres ejemplos los cuales se muestran a continuación con más detalle. (GitHub, 2018) [110]

5.3.1 Cartpole

Este ejemplo se desarrolla en una dimensión 2D y consiste en que el objeto de la imagen se mantenga verticalmente en equilibrio el mayor tiempo posible sin que se caiga. Este objeto está compuesto por una barra y un carro que se desplaza por un raíl de izquierda a derecha. El aprendizaje del sistema depende de cuatro componentes que son la velocidad y posición del carro, y la velocidad angular y el ángulo de la barra.

Al principio de la ejecución el tiempo que permanece de pie es mínimo, pero a medida que avanzan las simulaciones este valor del tiempo incrementa. De este modo el aprendizaje de las redes neuronales es más eficaz cuantas más simulaciones se realicen. Una puntuación de recompensa de 200 significa que el entorno ha sido dominado y por lo tanto el aprendizaje ha sido válido.

Cuando se ejecuta el programa se abren tres ventanas, en una se muestra el objeto en movimiento, en la segunda se muestra la imagen del objeto como un conjunto de píxeles y en la última se muestra una gráfica en la que se refleja la recompensa con respecto al número de la simulación realizada.

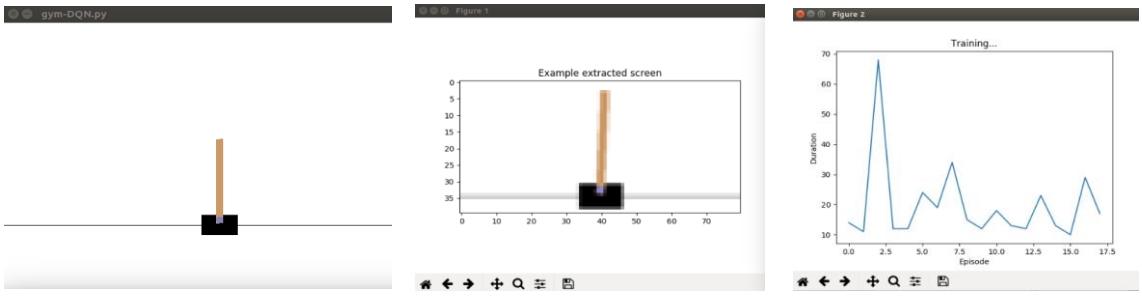


Figura 111: Venatanas de Cartpole

En la ventana del terminal aparece el número de la simulación y la recompensa correspondiente al tiempo que se mantiene en equilibrio. Las siguientes imágenes muestran el proceso de aprendizaje de Cartpole, en la imagen derecha se muestran los valores obtenidos en el terminal y la otra corresponde a su respectiva gráfica.

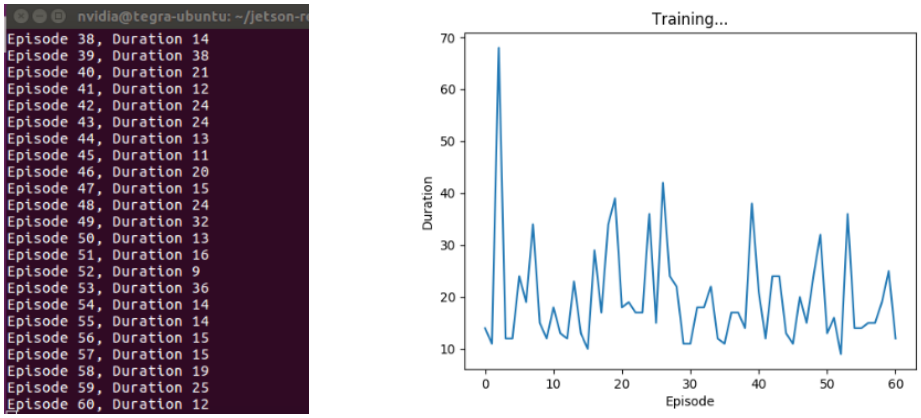


Figura 112: Cartpole Inicio del aprendizaje

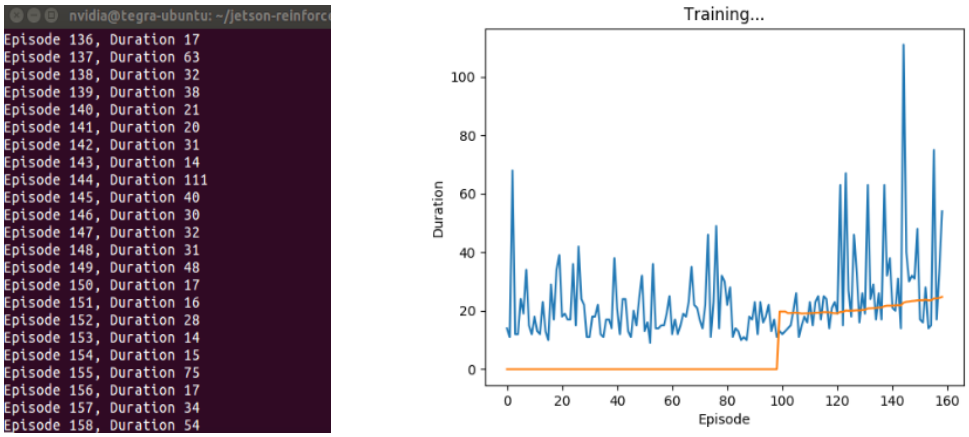


Figura 113: Cartpole Intermedio del aprendizaje

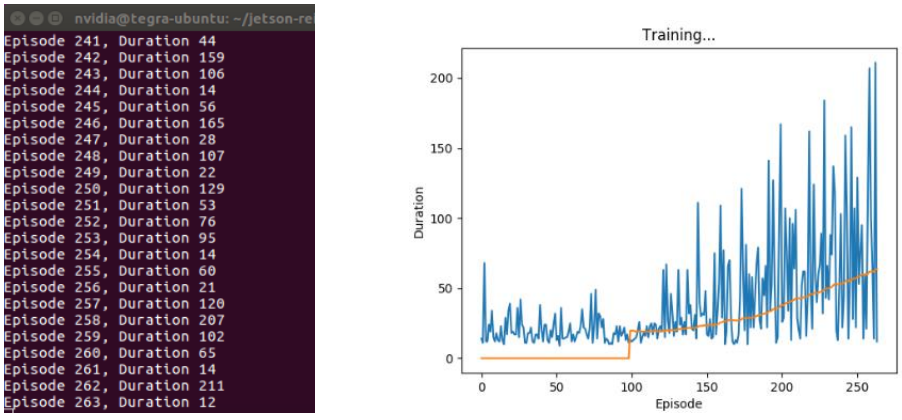


Figura 114: Cartpole Fin del aprendizaje

Se observa en la última imagen que fueron necesarias más de 250 simulaciones para controlar el entorno, siendo la simulación 258 cuando se llega al valor de 207. Se realizó un par de simulaciones más y se obtuvo el máximo valor en la simulación 262 con una puntuación de 211. La línea naranja de la gráfica señala la recompensa media, cuyos valores inferiores a 25 no son representados, por lo que a partir de la simulación cien aproximadamente ya se representa la línea, que es cuando alcanza este valor.

5.3.2 Lunar Lander

Este ejemplo simula el aterrizaje automático de una nave espacial, su objetivo es aterrizar lo más estable posible entre las dos banderas amarillas. Su funcionamiento consiste en cuatro acciones que son no hacer nada, desplazarse hacia la derecha, izquierda o hacia abajo. El aprendizaje depende de la velocidad de la nave y de sus coordenadas, por lo que la recompensa que se genere dependerá de estos dos parámetros. Cuanto menor sea la velocidad y más cerca se encuentre de las dos banderas, mayor será la recompensa. Al iniciar la simulación la nave se estrella muy bruscamente sobre la superficie y a medida que aumentan las simulaciones, el aterrizaje se realiza de forma más estable obteniendo así una mayor recompensa. Esto se debe a que los valores son actualizados en cada simulación, por lo que el entrenamiento toma los valores del episodio anterior e intenta aprender en base a ellos.

En las siguientes imágenes se muestra el proceso de aprendizaje de la nave, en el que se considera un aprendizaje válido si la recompensa es positiva y el promedio de la longitud es similar a 200. En este caso la simulación no se realizó hasta el final, debido a que esto supondría mucho tiempo. En la imagen derecha se muestran los valores obtenidos en el terminal y la segunda imagen es la simulación del aterrizaje de la nave.

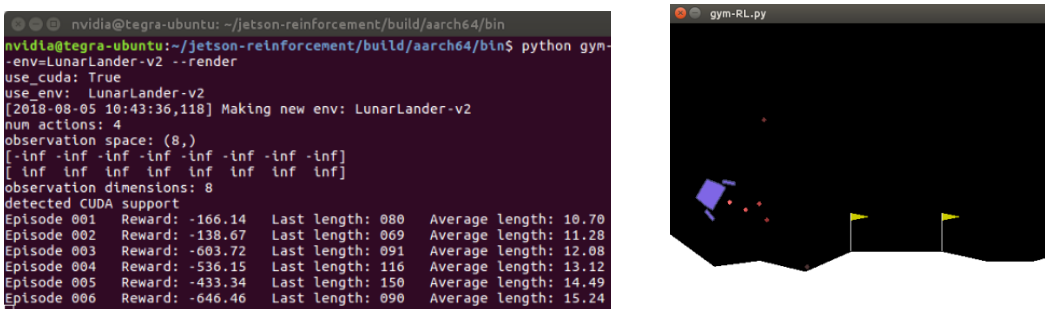


Figura 115: Lunar Lander Inicio del aprendizaje

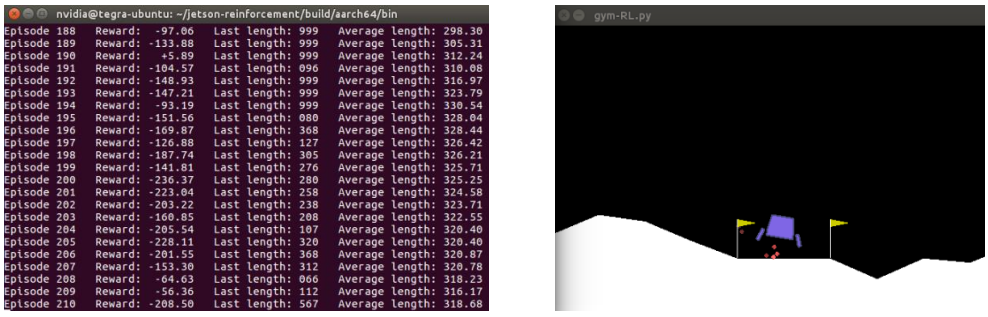


Figura 116: Lunar Lander Intermedio del aprendizaje

5.3.3 Brazo Robótico

Este ejemplo se desarrolla en el entorno Gazebo que es tridimensional y el objetivo es conseguir que la pinza del brazo robot toque al objeto con la mayor precisión posible. En este caso la base del brazo se encuentra fija y son las uniones o articulaciones entre las tres partes del robot las que se mueven y por tanto serán estas las que proporcionen el valor de la posición. La simulación del episodio finaliza cuando la pinza del robot toca el suelo, comenzando una nueva desde el punto inicial, que es el brazo del robot estirado hacía arriba. En este caso la recompensa depende de la distancia de la pinza del robot y el objeto, por lo que se considera positiva cuanta menos distancia exista entre ellos.

Cuando se ejecuta el programa se abre una ventana en la que aparece el brazo robot, un objeto y el agente que capta si el brazo interactúa con el objeto. El brazo robot se dirigirá hacía el objeto para tocarlo, realizando al principio movimientos imprecisos. A medida que aumentan las simulaciones los movimientos son cada vez mejores en los que llega a tocar el objeto con mayor precisión. Se ha de tener en cuenta que el objeto puede ser movido por el usuario en cualquier momento de la simulación, por lo que el brazo robot ha de cambiar las coordenadas que antes le eran objetivas e ir a tocar al objeto en su nueva posición.

En las siguientes imágenes se muestra el proceso de entrenamiento.

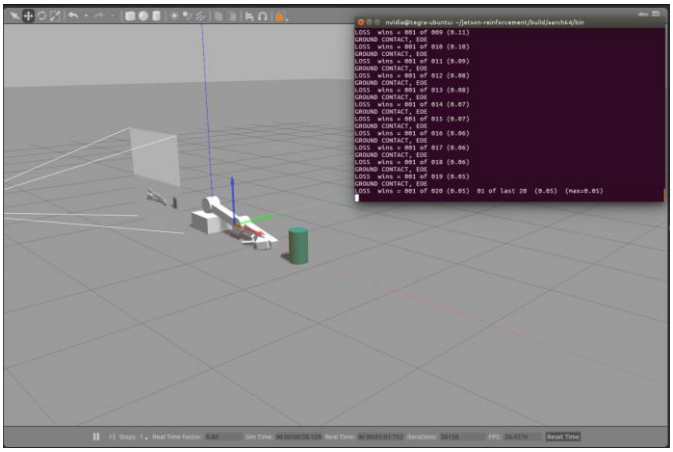


Figura 117: Brazo Robótico Inicio del aprendizaje

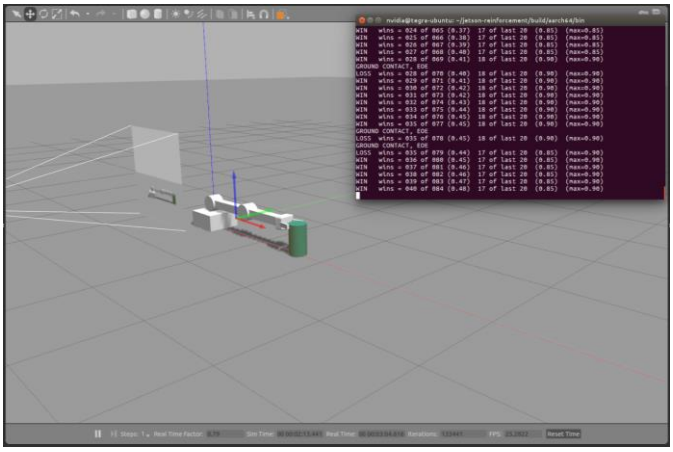


Figura 118: Brazo Robótico Intermedio del aprendizaje

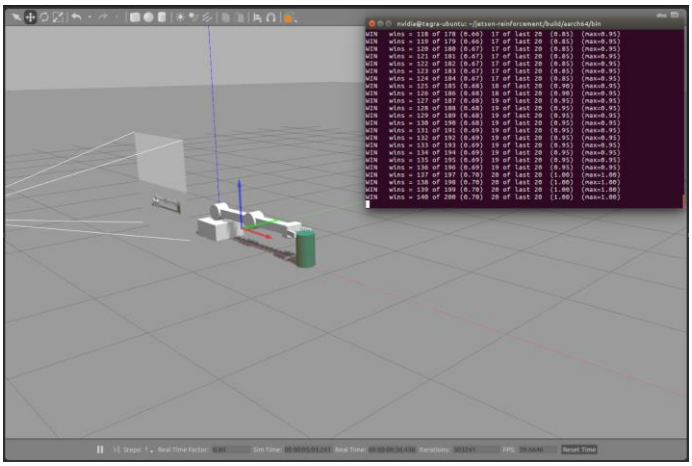


Figura 119: Brazo Robótico Fin del aprendizaje

5.4.OpenCV

Esta biblioteca OpenCV se basa en el procesamiento y análisis de imágenes por ordenador, por lo que es muy utilizada en aplicaciones de reconocimiento, detección y filtrado de imágenes. La versión que se adquirió con el JetPack fue la 2.4.13, por lo que posteriormente se instaló la versión 3.4.1 (GitHub, 2018) [111]. Para su instalación se siguió el tutorial de (Build OpenCV on NVIDIA Jetson TX2, 2017) [101] y (Build OpenCV on NVIDIA Jetson TX2, 2018) [102] , por lo que se descargó desde la plataforma Github, pero se obtuvo un problema cuando se trabajó con el programa YOLOv3. Dicho programa detecta objetos en una imagen y tiene tres modos de funcionamiento, una de ella es la cámara, la otra es la detección por vídeo y el tercer modo es mediante una imagen. En un primer momento todo funcionó correctamente y se realizaron simulaciones, sin embargo, el problema se obtuvo al ejecutar el modo de imagen del programa YOLOv3. Este se ejecutaba y sacaba los resultados correctamente por el terminal, pero no generaba la imagen con los resultados de detección obtenidos. Esto se debe a que existen errores en la API de esta versión, por lo que se ha de descargar otra versión anterior, por ello se instaló la 3.4.0 siguiendo el tutorial de (Jkjung-avt.github.io, 2018) [112]

5.4.1 Conversión RGB a gris, Suavizado y Detector Canny

Las imágenes de izquierda a derecha que se muestran a continuación representan la conversión de RGB a tonos de grises, la siguiente un suavizado o desenfoque Gaussiano y la tercera el detector de bordes Canny. La imagen fue tomada con la cámara de la tarjeta TX2 y estas pruebas se pudieron realizar con ambas versiones.

Figura 120: Conversión RGB a gris, Suavizado y Detector Canny

5.4.2 Detector de objetos de un determinado color

Este programa consiste en detectar los objetos que se encuentran en una habitación de un color determinado. Está basado en OpenCV y su lenguaje de programación es Python, para su ejecución se utilizó la cámara de la tarjeta Jetson TX2. Para realizar su programación se consultó la página de OpenCV (Opencv.orgn n. d.) [78] y diversos foros.

Su funcionamiento consiste en transformar la imagen en color captada por la cámara a un formato HSV. Al realizar esta transformación se ha de indicar con que gama de colores se va a trabajar, en este caso se indicó el color azul. Por lo que se indican seis parámetros, los tres primeros corresponden al color desde el cual va a empezar a captar, y los otros tres señalan el límite del rango. Para eliminar cualquier posible ruido de la imagen se le aplica el proceso morfológico de apertura, por lo que se realiza la combinación de erosión y dilatación.

Figura 121: Imagen HSV con ruido

Para ello se determina el kernel que se les aplicará a los píxeles de la imagen, en este caso es una matriz de 10x10. Por lo que primero se le somete a la erosión y después a la dilatación, cuya salida se muestra en la siguiente imagen, donde se observa que el ruido de la imagen ha desaparecido.

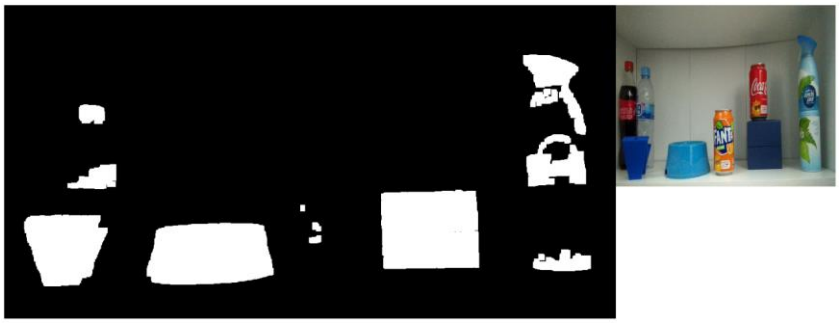


Figura 122: Imagen HSV sin ruido

Con el objeto ya detectado se le aplica una función especial que determina el contorno de este, y guarda en una variable el conjunto de todos los puntos del contorno detectado. De manera opcional se la cambió el color de la parte que la cámara muestra del objeto detectado, por lo que los píxeles de esta zona serán de color gris en este caso.

Mediante un bucle *for* y los puntos del contorno detectado encuentra el cuadro delimitador del objeto y lo dibuja.

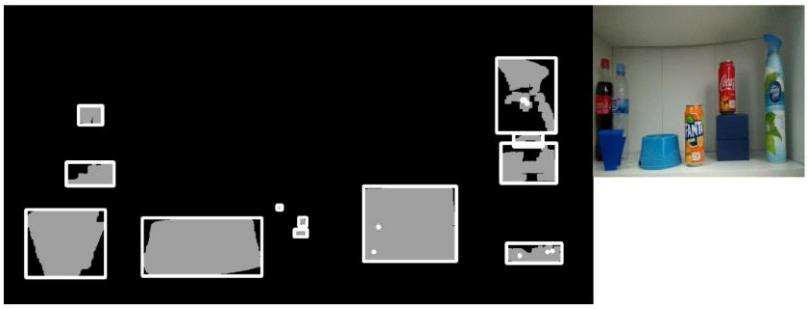


Figura 123: Objetos con cuadros delimitadores

Posteriormente se transforman los datos del eje x e y, a formato de cadena, para ser mostrados en la pantalla. Como se puede observar aparece una segunda ventana en la que se muestra la imagen sin procesar para comprobar de este modo que la cámara detecta correctamente.

Figura 124: Coordenadas del objeto

En la parte superior izquierda se muestran las coordenadas del eje X e Y donde se encuentra el centro del cuadrado delimitador. En este caso está situado a 426 píxeles en el eje horizontal y a 441 en el eje vertical, siendo el tamaño de la imagen 1280x720.

5.4.3 Detector de circunferencias

El siguiente programa que se muestra consiste en detectar las circunferencias que aparecen en la imagen. Al igual que el anterior se ha realizado en base a OpenCV y su ejecución se llevará a cabo mediante la cámara.

El programa se inicia transformando la imagen de color a escala de grises y posteriormente se le aplica una función especial que detecta si hay alguna circunferencia en la imagen. En esta función se le ha de indicar que tamaño de circunferencias se desea que capte la cámara, mediante el valor máximo y mínimo de los radios.

En el caso de que detecte alguna circunferencia guarda los valores captados en una variable, la cual se utiliza posteriormente en un ciclo *for* para dibujar la circunferencia. Esto se lleva a cabo mediante una función que cuenta con tres parámetros a los que se les pasa el valor del eje (x, y) y el de su radio.

Figura 125: Circunferencias detectadas

Se añadió al código la opción de que las circunferencias sean clasificadas según el tamaño de su radio, por lo que se determina un valor y se utiliza un ciclo *if*. Si el radio captado por la cámara superaba el valor del umbral, mostrará que es mayor, en caso contario señalará que no. Finalmente se muestra la imagen con la circunferencia dibujada, y se muestra por pantalla el valor de su radio y si supera el umbral establecido.

Figura 126: Muestra de los datos de la circunferencia

5.5. YOLO

Existen diferentes algoritmos de visión artificial como RetinaNet, SSD y YOLO, cuyo funcionamiento se basa en detectar y clasificar objetos. Las pruebas que se realizaron con la tarjeta Jetson TX2 se basan en el algoritmo YOLOv3 que fue instalado como se indica en los siguientes tutoriales (Package OpenCV Part 2 and YOLO!, 2018) [104] (yolov3 custom object detection in linux mint or ubantu, 2018) [107] y (Jkjung-avt.github.io, 2018) [113]. A continuación, se realiza una breve introducción de YOLO en la que se explica cómo ha variado su modo de funcionamiento entre las distintas versiones, para ello se tomó como referencia (Medium, 2018) [55] (You Only Look Once: Unified, Real-Time Object Detection, 2016) [97] y (Santos, 2018) [87]

5.5.1 YOLOv1

La red neuronal de la primera versión de YOLO que se presentó cuenta con un total de 26 capas, de las cuales las 24 primeras son capas convolucionales y las otras dos están conectadas linealmente. La imagen a la entrada presenta un tamaño de (224 x 224), pero para poder detectar mayores detalles de la imagen se aumenta su resolución a (448 x 448). La cual posteriormente se reduce para obtener una salida de (7 x 7) mediante capas de convolución que a su vez algunas de ellas utilizan capas de reducción 1x1 para reducir la profundidad de la imagen.

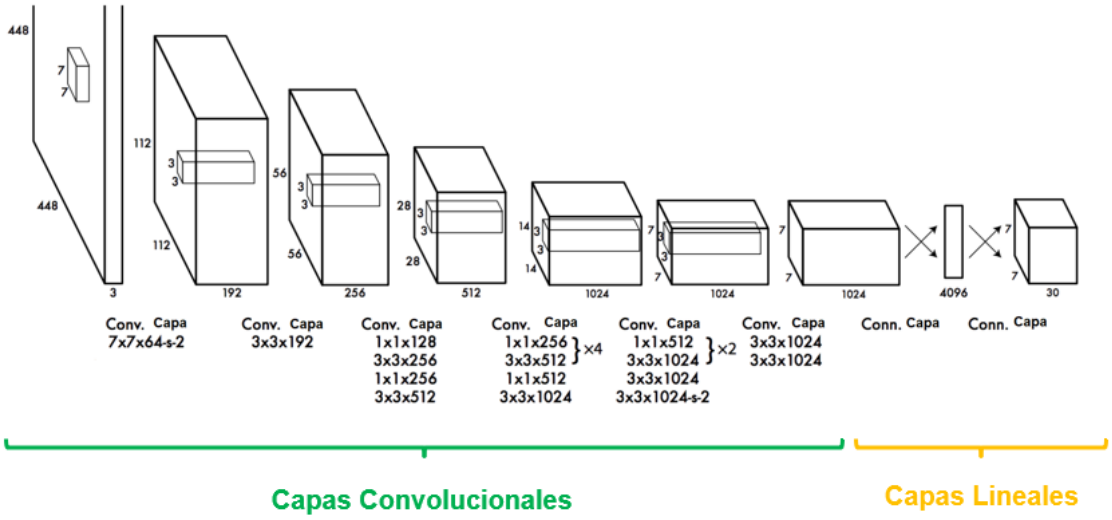


Figura 127: Red neuronal con 26 capas

De este modo tras pasar todas las capas la imagen queda dividida en un total de 49 celdas, que se expresa como (*7 X 7*). Debido a la conexión de regresión lineal de sus dos últimas capas cada una de estas celdas predice dos *boxes bounding* y se ha de tener en cuenta que cada celda de la imagen detecta sólo un objeto.

Por ejemplo, en la imagen inferior la celda está señalada de amarillo y detectará a la persona, mientras que los *boxes bounding,* son los señalados en color rojo.

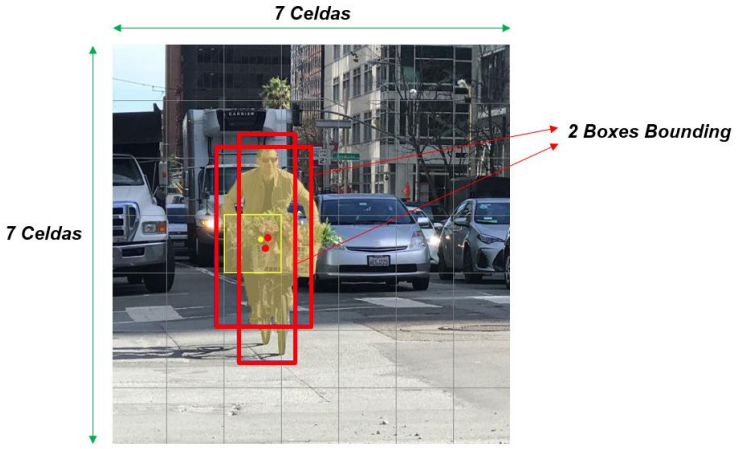


Figura 128: YOLOv1 (7x7)

El hecho de que cada celda de la imagen sólo detecte un objeto limita el número de cosas detectadas por YOLO. Por lo que, si los objetos se encuentran muy cerca, algunos de estos no serán detectados debido a esta limitación. Ejemplo de esto se muestra en la siguiente imagen, en la que aparecen un total de nueve personas en la parte inferior izquierda, pero sólo son detectadas cinco debido a la proximidad que hay entre ellas.

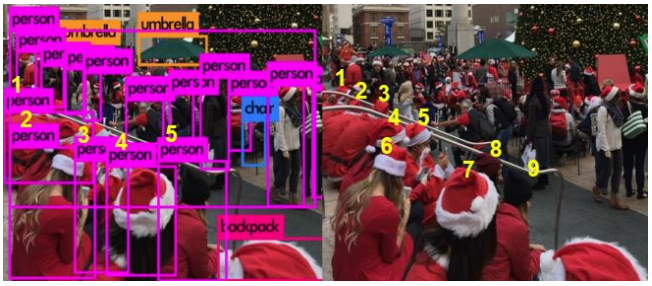


Figura 129: Limitación en la detección

Cada *box bounding* genera cinco datos de los cuales el valor de cuatro de ellos varía entre cero y uno, y son los siguientes:

- X: Coordenada X del centro del *box bounding*.
- Y: Coordenada Y del centro *box bounding*.
- H: Representan la altura.
- W: Representa la anchura.

El quinto dato hace referencia a la probabilidad de que la celda contenga alguna clase de objeto.

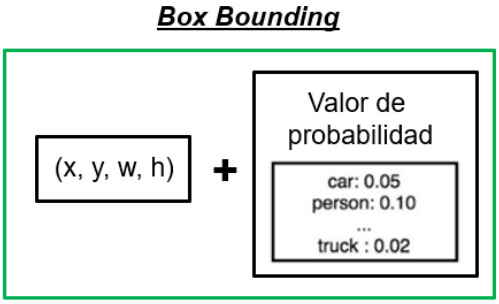


Figura 130: Componentes del Box Bounding

De este modo se determina el valor del Tensort, el cual se expresa mediante la ecuación:

$$Tensort = S \cdot S \cdot (B \cdot 5 + C)$$

Donde B indica el número de *boxes bounding* que tiene cada celda, en este caso dos; mientras que el número 5 hace referencia a los cinco datos que tiene cada *box bounding* y la C señala las clases de objeto que es capaz de detectar, que en esta versión son 20, lo cual se muestra en anexos.

El entrenamiento se inicia calculando el valor de la probabilidad (*Box Confidence Score*) de que exista un objeto en la celda, este depende de la intersección entre el objeto y la imagen y se expresa mediante la siguiente expresión.

$$Box\ Confidence\ Score = P_r(object) \cdot IoU$$

Se ha de tener en cuenta que, aunque cada celda genere dos *boxes bounding,* sólo una de ellas se utiliza para la detección de objetos, y se selecciona aquella que tenga mayor valor IoU. Reduciendo de este modo el número de *boxes bounding,* ya que sólo se analizarán aquellos que son responsables de esta detección. El cuadro verde indica el límite real del objeto y el rojo al *box bounding.*

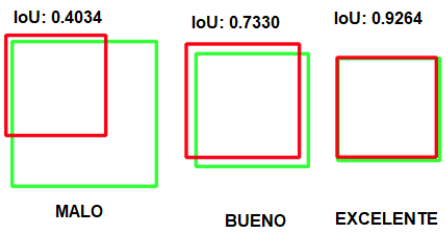


Figura 131: Valores IoU

El valor de probabilidad de que se encuentre un objeto en la celda se multiplica por el total de las clases de objetos, obteniendo como resultado el valor de la probabilidad de cada clase (*Class Confidence Score*), la cual se expresa mediante la ecuación:

$$P_r(class_i) \cdot IoU = (P_r(object) \cdot IoU) \cdot P_r(class_i/object)$$

Una vez determinados estos valores se determinan las pérdidas existentes, que se dividen en tres tipos.

Pérdida de clasificación

Al ser las dos últimas capas de la red neuronal de regresión lineal a estas se les aplica una función lineal en la clasificación de las celdas. Mientras que en el resto de las capas se utiliza una función lineal rectificada con pérdidas. En la función lineal la salida depende directamente de la entrada y sus valores no están limitados, su ecuación es:

$$f(x) = x$$

Siendo la función lineal rectificada (You Only Look Once: Unified, Real-Time Object Detection, 2016) [97] utilizada por el resto de las capas:

$$f(x) = \begin{cases} x, & Si\ x > 0 \\ 0.1x, & otros \end{cases}$$

Estos valores son optimizados mediante el modelo de pérdida de clasificación, que hace referencia al error de probabilidad de que el objeto pertenezca a una clase y se obtiene en función de:

$$Pérdida_{Clasificación} = \sum_{i=0}^{S^2} 1_i^{obj} \sum_{c\ \epsilon\ clase} ((p_i(c) - \hat{p}_i(c))^2$$

Cuyos parámetros hacen referencia a:

- S: Es la dimensión del tensor, en este caso es 7.
- 1_i^{obj}: Este parámetro es uno si la celda contiene al objeto, en caso contrario su valor es cero.
- $p_i(c)$: Este valor es generado por la red neuronal y señala la probabilidad de que el objeto de la celda i pertenezca a una clase C. Se obtiene de las funciones lineales y lineales rectificadas con pérdidas.
- $\hat{p}_i(c)$: Indica la probabilidad que existe de que esa clase C se encuentre en la celda i.

Pérdida de seguridad

Indica el error de probabilidad de que la celda contenga alguna clase de objeto, este valor se calcula en función de:

$$Pérdida_{Seguridad} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C}_i)^2$$

Cuyos parámetros hacen referencia a:

- S: Es la dimensión del tensor, en este caso es 7.
- B: El número de *boxes bounding,* en esta versión son 2.
- 1_{ij}^{obj}: Este parámetro es uno si el *box bounding* de la celda es el encargado de detectar al objeto, en caso contrario su valor es cero.

- C_i: Es el valor generado por la red neuronal que indica la probabilidad existente a que el objeto se encuentre en esa celda.
- $\hat{C}_i$: Indica el valor de seguridad de que esa clase de objeto se encuentre en la celda i.
- λ_{noobj}: Su valor es 0.5 y reduce la pérdida cuando no se detectan objetos en una celda.
- 1_{ij}^{noobj}: Es el complemento de 1_{ij}^{obj}.

Pérdida de localización

Determina los errores según las ubicaciones y tamaños de los *boxes bounding* utilizados para la detección de los objetos. Como ya se ha dicho anteriormente solo se utiliza uno de ellos, el de mayor valor IoU. La expresión que determina esta pérdida es:

$$Pérdida_{Loc} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right]$$

Cuyos parámetros hacen referencia a:

- S: Es la dimensión del tensor, en este caso es 7.
- B: El número de *boxes bounding,* en esta versión son 2.
- 1_{ij}^{obj}: Este parámetro es uno si el *box bounding* de la celda es el encargado de detectar al objeto, en caso contrario su valor es cero.
- X, Y, w, h: Indican las coordenadas del *box bounding.*
- λ_{coord}: Este valor es 5 e incrementa el valor del peso de pérdida en las coordenadas de los *boxes bounding* límites.

Para evitar que un objeto sea predicho y señalado por varios recuadros se aplica el método NMS (*Non-Maxima Suppresion*) que consiste en ordenar los valores obtenidos del entrenamiento de la red neuronal y quedarse con los valores más altos. Esto se observa en la primera imagen, en la que se muestran los *boxes bounding* de mayor valor.

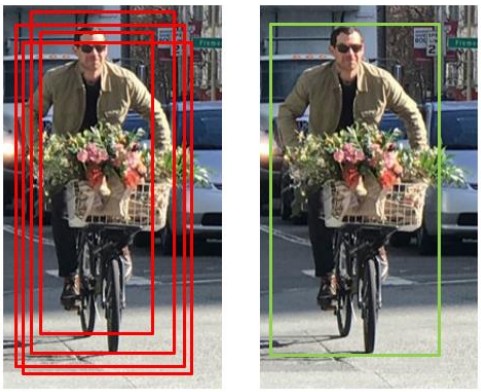


Figura 132: Non-Maxima Suppresion

Posteriormente se toma el valor más alto por lo que se eliminan aquellos que pertenezcan a la misma clase y tengan un valor de IoU superior a 0.5, es decir se obtiene así un único *box bounding,* aquel de mayor valor.

Al aplicar la *Non-Maxima Suppresion* se incrementa el valor de la precisión promedia (mAP) en un 2 a 3 %. Este parámetro indica la media de las predicciones máximas obtenidas (P) a diferentes valores de *Recall* (R). Este último parámetro indica como de favorable son las predicciones que se han obtenido con respecto a todos los casos e incrementa a medida que se aumentan las predicciones favorables.

Sus valores se obtienen aplicando las siguientes ecuaciones:

$$P = \frac{TP}{(TP + FP)} \qquad R = \frac{TP}{(TP + FN)}$$

Donde las siglas significan lo siguiente:

- TP: Verdadero positivo.
- FP: Falso positivo.
- TN: Verdadero Negativo.
- FN: Falso Negativo.

Para aclarar mejor esta explicación se plantea un pequeño ejemplo en el que se determinaran los valores de (P) y (R), con cuyos valores se determina la gráfica de precisión respecto a *Recall*. Se ha tomado como referencia para esta información (Medium, 2018) [56], aunque el ejemplo realizado es propio.

El ejemplo que se plantea es que tras el entrenamiento de la red neuronal se ha obtenido diez predicciones del objeto "perro", las cuales se han ordenado de mayor a menor según el nivel de probabilidad predicho. Además, se ha señalado si la predicción obtenida es falsa o verdadera, en función si ha superado el valor IoU de 0.5. Una vez organizados estos datos se ha calculado los valores de la precisión y *Recall*, en función de las ecuaciones superiores. Todo esto se muestra en la siguiente tabla y los cálculos realizados se encuentran en anexos.

Predicciones obtenidas de "Perro"			
Ranking	¿Es valida?	Precisión	*Recall*
1	Verdadera	1	0.2
2	Verdadera	1	0.4
3	Verdadera	1	0.6
4	Falsa	0.75	0.6
5	Falsa	0.6	0.6
6	Verdadera	0.67	0.8
7	Falsa	0.58	0.8
8	Falsa	0.5	0.8
9	Verdadera	0.51	1
10	Falsa	0.6	1

Tabla 23: Precisión-Recall

Con estos valores obtenidos se determina la gráfica que se muestra a continuación:

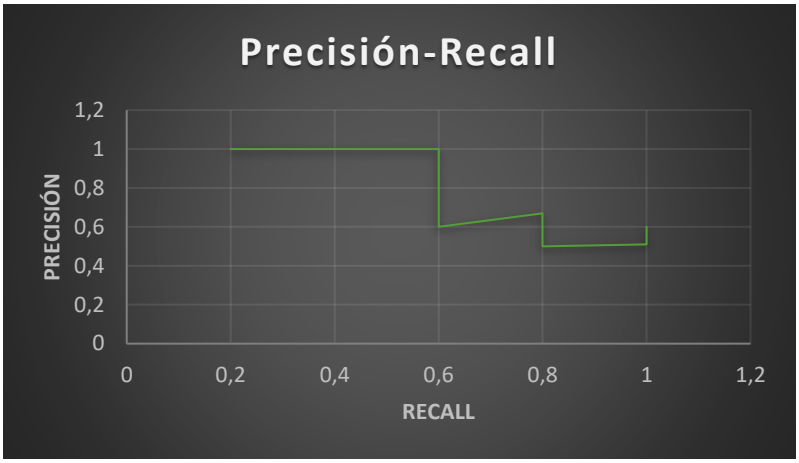


Figura 133: Gráfica Precisión-Recall

En base a esta gráfica se determina la precisión media (AP) de las predicciones obtenidas, cuyo valor corresponde al área que se encuentra debajo de la gráfica verde. Pero como se observa en la imagen, existen picos en la gráfica por lo que se ha de suavizar para obtener de este modo un valor más preciso. Determinando de este modo la curva P-R interpolada que se determina en función del valor de precisión máxima para cualquier calor de *Recall*.

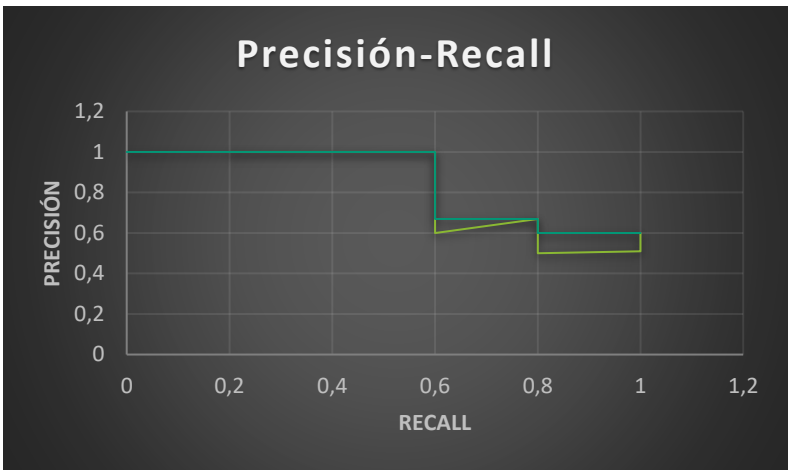


Figura 134: Gráfica Precisión Máxima-Recall

Los valores que se muestran en la siguiente tabla corresponden a la precisión máxima, que han sido tomados de la gráfica anterior.

Recall	Precisión Máxima
0	1
0.1	1
0.2	1
0.3	1
0.4	1
0.5	1
0.6	0.67
0.7	0.67
0.8	0.67
0.9	0.6
1	0.6

Tabla 24: Precisión Máxima-Recall

Obteniendo de este modo el valor de la AP que corresponde al área que se encuentra debajo de la gráfica verde más oscura, cuyo valor numérico se determina como la suma total de cada precisión máxima por su *Recall* correspondiente divido entre once, que corresponde a los números de valores de *Recall* tomados.

$$AP = \frac{1 \cdot 6 + 0.67 \cdot 3 + 0.6 \cdot 2}{11} = 0.84$$

Finalmente, para determinar el valor de mAP, se ha de dividir la suma de todas las AP entre el número total de clases. La siguiente imagen muestra un ejemplo de esto.

$$mAP = \frac{\sum AP_{Clases}}{Número\ Total_{Clases}}$$

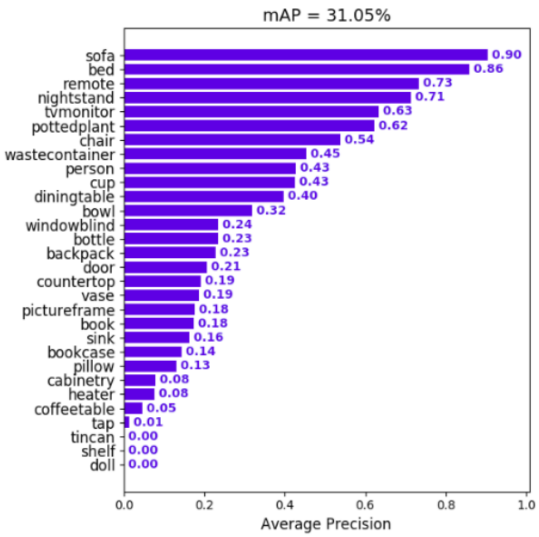


Figura 135: Gráfico de la precisión promedia

5.5.2 YOLOv2

Esta versión de YOLO se basa en Darknet-19 que está formada por un total de 30 capas, de las cuales once se utilizan para la clasificación y el resto para la detección de objetos (Towards Data Science, 2018) [94]. Su entrenamiento se divide en dos fases, en la que empieza clasificando los objetos mediante imágenes de tamaño (*224 x 224)* celdas y posteriormente se realiza la clasificación con (448 x 448), mejorando de este modo un 4 % el valor de mAP.

La mejora que presenta YOLOv2 con respecto a la versión anterior es que los *boxes bounding* no son predichos por la red neuronal, sino que se determinan al inicio del entrenamiento cinco *boxes bounding* aleatorios por cada objeto. Por lo que, si cada celda contiene cinco *boxes bounding* y estos a la vez contienen cinco parámetros y 20 clases de objetos, como la versión anterior, entonces cada celda cuenta con un total de 125 parámetros.

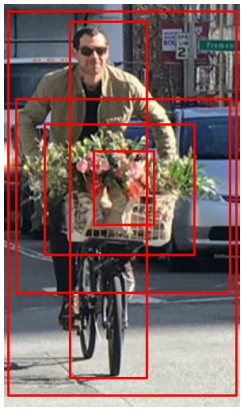


Figura 136: Cinco Boxes Bounding

Por ello se eliminan las dos últimas capas que estaban conectadas de forma lineal, ya que ahora los *boxes bounding* no son predichos por la red neuronal. Otra modificación que realiza es cambiar el tamaño de la imagen a (416 x 416), de este modo el número de celdas en la que queda dividida la imagen representa un número impar a cada lado. Con esto se consigue que el entrenamiento parta de una celda, la cual suele ser la del centro de la imagen, ya que en la mayoría de las fotos el objeto se encuentra en el centro. De este modo el entrenamiento se inicia de manera más estable tomando como referencia esta celda y en base a los cinco *boxes bounding.* En la siguiente imagen se muestra la diferencia entre un número par e impar de celdas en una imagen.

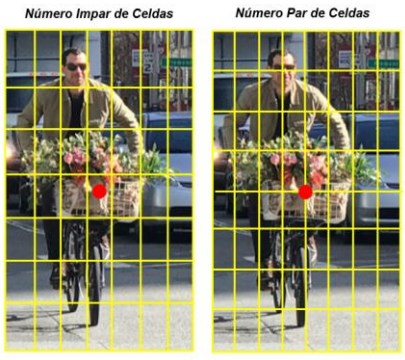


Figura 137: Diferencia entre número par e impar de cledas

Al utilizar más números de *boxes bounding* aumenta las posibilidades de detectar los objetos en la imagen y el valor de *Recall* aumenta, pero sin embargo el valor de mAP se ve reducido.

Los parámetros generados por cada *box bounding* son los siguientes:

$$b_x = \sigma(t_x) + C_x$$

$$b_y = \sigma(t_y) + C_y$$

$$b_w = p_w \cdot e^{t_w}$$

$$b_h = p_h \cdot e^{t_h}$$

Donde b_x y b_y hacen referencia a las coordenadas del centro, y b_w y b_h al ancho y alto del *box bounding.* Estos dos últimos valores se obtienen multiplicando la transformación del desplazamiento del registro por el ancho y alto del cuadro real del objeto (p_w y p_h). Los parámetros t_x, t_y, t_h y t_w son los datos obtenidos por la red neuronal, y C_x y C_y corresponden a la distancia con respecto a la celda de la esquina superior izquierda.

Como se observa en la ecuación superior las coordenadas b_x y b_y se obtienen mediante una función sigmoide, por lo que el resultado de la salida varía entre cero y uno. Esto se aplica para limitar el valor en el caso de que las coordenadas t_x y t_y sean mayores a uno, lo que indicaría que el centro se encuentra fuera de la celda. Este valor no representa una coordenada absoluta, sino que hace referencia a una compensación que es relativa a la esquina superior izquierda.

En la imagen que se muestra a continuación el recuadro rojo hace referencia a los límites reales del objeto y el naranja al *box bounding* predicho.

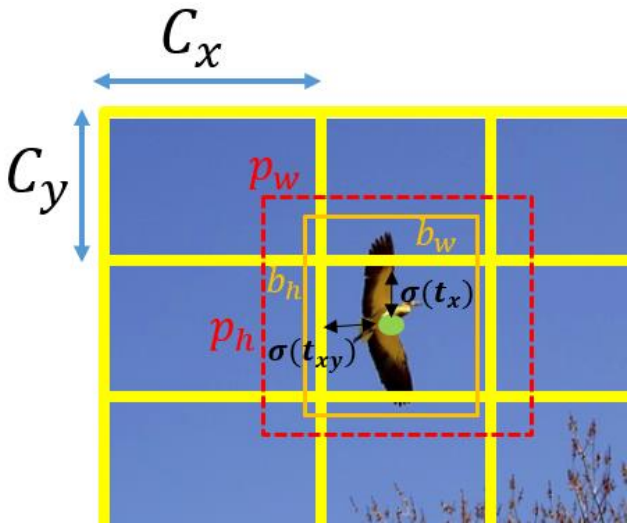


Figura 138: Cuadro delimitador

Al igual que la versión anterior se utilizan capaz convolucionales que reducen el tamaño de la imagen, generando pérdida en la resolución, por lo que resulta más complicado detectar objetos pequeños en la imagen. Por lo que YOLOv2 aplica el método *passthrough* que consiste en concatenar capas y aplicar filtros de convolución para realizar las predicciones. Tras pasar todas las capas la imagen queda dividida en celdas de (*13 X 13*) en lugar de (*7 X 7*), por lo que la imagen de entrada puede ser de cualquier tamaño entre 320 como mínimo y 608 máximo, siempre que este sea múltiplo de 32. Esto se debe a que la red neuronal cambia aleatoriamente el tamaño

de la imagen cada diez lotes. De este modo el entrenamiento se realiza a diferentes escalas siendo más preciso y exacto el entrenamiento. Esto se puede habilitar o deshabilitar mediante el comando *Random*.

Otra mejora que incorpora YOLOv2 es la estructura *WordTree* para determinar la clasificación de las etiquetas. De este modo ahora las etiquetas no son excluyentes, por lo que, si el algoritmo no es capaz de distinguir por ejemplo el tipo de raza de un perro, este será detectado como perro, y no según la raza a la que pertenece.

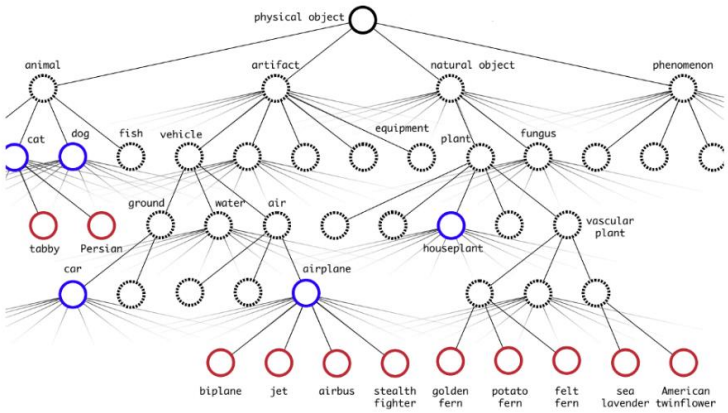


Figura 139: Estructura WordTree

Esta estructura está formada por 1000 nodos con etiquetas específicas y por 369 de clases generales. El valor de la probabilidad de que pertenezca a una clase determinada se realiza mediante múltiples funciones *softmax* mediante las cuales se determinan las subclases de cada clase principal, y convierte los valores obtenidos en probabilidades que sumen hasta un máximo de uno, que será el valor de la clase general.

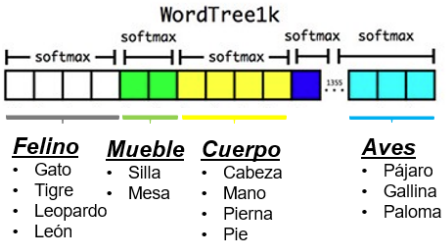


Figura 140: Funciones Softmax

A continuación, se muestra un ejemplo de cómo se determina la probabilidad de clase, este ejemplo se basa en la raza de perro "bóxer".

$$P_r(bóxer) = P_r(bóxer|perro) \cdot P_r(perro|animal) \cdot P_r(animal|mamífero) \cdot P_r(mamífero|ser\ vivo)$$

En la que $P_r(mamífero|ser\ vivo)$ tendría el valor de 1 y YOLOv2 recorre el camino con probabilidad más alta hasta llegar aquel nodo donde se supere el umbral de probabilidad de clase establecido.

5.5.3 YOLOv3

Esta versión a diferencia de la anterior es más precisa en la detección de objetos pequeños y puede alcanzar unos 30 FPS con una Titan X. Se basa en Darknet-53 por lo que cuenta con un total de 106 capas, de las cuales 53 se utilizan para la clasificación y el resto para la detección de los objetos. Por lo que al estar formada por más capas hace que sea más lento, aunque es mucho más preciso en la detección y clasificación de objetos. (Towards Data Science, 2018) [95] (YOLOv3: An Incremental Improvement, n. d) [96]

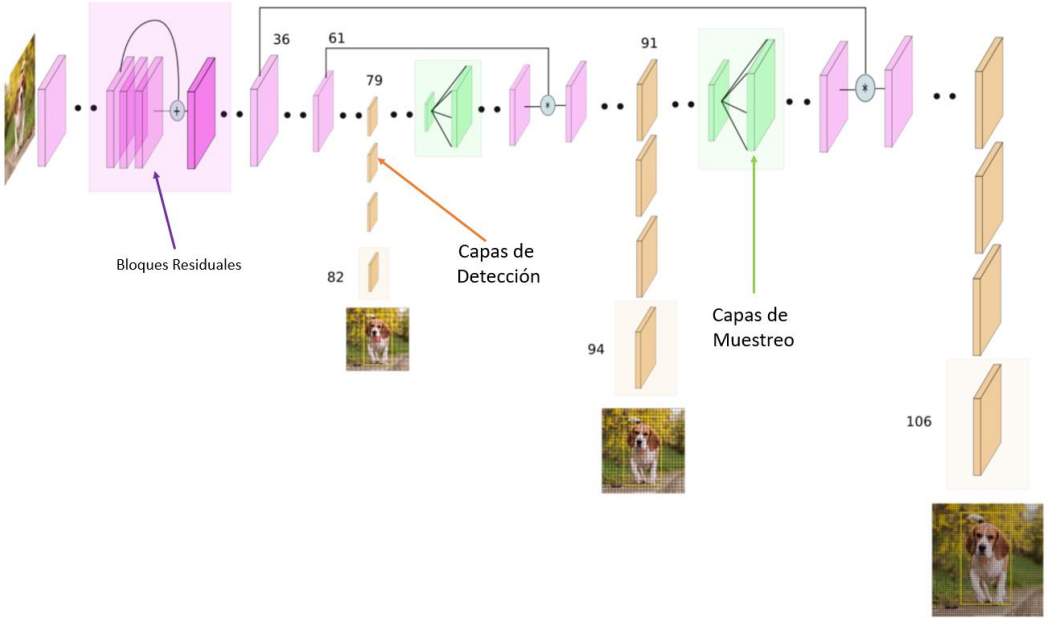


Figura 141: Estructura de las capas YOLOv3

La imagen superior muestra la estructura que presenta el conjunto de las capas convolucionales de la red neuronal. Esta versión se caracteriza por realizar la detección de las características en tres mapas de diferentes tamaños colocados en tres posiciones distintas en la red. El factor que reduce el tamaño de las imágenes se denomina *stride* cuyos valores son 32, 16 y 8. El aprendizaje de la red neuronal se divide en tres fases que están señaladas por los cambios de tamaño de la imagen.

En la primera fase durante las primeras 78 capas los bloques residuales copian y optimizan la imagen original. En la capa 79 la imagen se somete a cuatro capas de detección, en las que aprende las características del objeto y las clasifica según la probabilidad de clase y según las coordenadas de los *boxes bounding*. Y es al pasar de la capa 81 a la 82 donde se aplica el primer *stride* de valor 32.

Posteriormente la imagen es sometida a capas de muestreo ascendente cuyo factor es dos, lo que permite que la red aprenda aquellas características necesarias para detectar objetos pequeños. Estas capas se concatenan con la capa anterior 61 y se les someten a bloques residuales. En la capa 91 se le vuelve a aplicar capas de detección 1x1 hasta la capa 93, y es en esta donde se aplica el segundo *stride* de valor 16 en este caso.

Nuevamente se le aplica unas capas de muestreo ascendente y se le concatena con la capa 36 obteniendo así las imágenes o mapas con características combinadas. Estos son sometidos a

bloques residuales para agrupar esta concatenación de capas y en la capa 105 se le vuelve a someter a capas de detección obteniendo en la capa 106 la imagen reducida por el factor 8.

En YOLOv3 a diferencia de la versión anterior se generan tres *boxes bounding* en cada celda, aunque sólo uno de ellos se utiliza para predecir la etiqueta del objeto, por lo que se toma aquel que tenga el valor IoU más alto. Como cada celda predice tres *boxes bounding* y se utilizan tres escalas diferentes esto hace que sean necesarios 9 *boxes bounding* que son diferentes según la escala en la que se utilice. El número de estos, el uso de tres escalas diferentes y la concatenación con capas anteriores es lo que permite a YOLOv3 aumentar la precisión en la detección de objetos pequeños. El Tensort se obtiene aplicando la siguiente ecuación:

$$Tessort = (S \cdot S) \cdot B \cdot (5 + C)$$

En el que B representa al número de cuadros delimitadores que hay en una celda, en este caso es tres. El número cinco hace referencia a (t_x, t_y, t_w, t_h) y al valor de objetividad, mientras que C señala el número de clases que la red va a detectar. La lista de estas clases se encuentra en anexos.

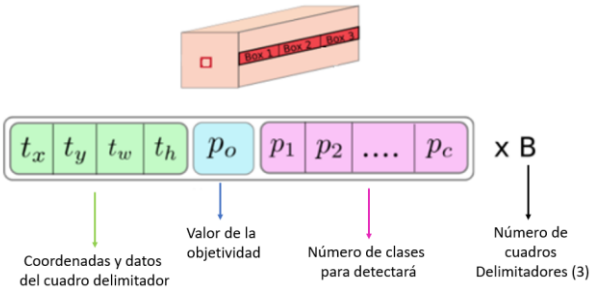


Figura 142: Profundidad de la celda

Otra de las novedades que incorpora YOLOv3 es que la clasificación de los objetos no las realiza con funciones *sofmax* sino mediante clasificadores logísticos. Estos valores pueden ser superiores a uno y para su cálculo se utiliza la pérdida de entropía cruzada en cada etiqueta, en vez del error de pérdida de clasificación. Con este método se obtiene la clasificación de las etiquetas mediante el cálculo de la probabilidad a que pertenezca a esa clase, la cual se determina mediante la ecuación. (labels?, 2018) [45]

$$q(y|X) = \begin{cases} c(x), & Si\ y = 1 \\ 1 - c(x), & Si\ y = 0 \end{cases}$$

Mientras que la detección se realiza en función de un valor de seguridad, en el que si el *box bounding* no supera el valor del umbral es descartado. Este valor varía desde cero a uno y hace referencia a la probabilidad de que el objeto este contenido dentro del box bounding predicho. Por lo que se considera un uno a la celda que se encuentre en el centro del cuadro límite real del objeto y un cero a las celdas que se encuentran alejadas del objeto. Al igual que en las versiones anteriores se le aplica el método *Non-Maxima Suppresion* para quedarse con el *box bounding* de mayor valor y evitar así que un objeto sea detectado por múltiples *boxes bounding*.

En la siguiente tabla se muestran los valores de precisión media de algunos de los algoritmos de visión artificial más utilizados (YOLOv3: An Incremental Improvement, n.d). [96]

- AP_{50}: Hace referencia al valor de la precisión media considerando como valor de *IoU* 0.5, por lo que si la intersección entre el *box bounding* predicho y el cuadro limite real del objeto no es superior al 50%, este se da como un falso positivo.
- AP_{75}: En este caso el valor de *IoU* es 0.75 y se realiza de modo similar que, en el caso anterior, considerando como valor válido si la intersección es superior al 75%.

Algoritmo	Estructura	Precisión Media (AP)	AP_{50}	AP_{75}
YOLOv3-608	Darknet-53	33	57,9	34,4
YOLOv2	Darknet-19	21,6	44	19,2
RetinaNet	ResNet-101-FPN	39,1	59,1	42,3
SSD	ResNet-101-SSD	31,3	50,4	33,3

Tabla 25: Valores de AP

En la tabla se observa que el algoritmo más eficiente es *RetinaNet* en el que sus valores de precisión son muy elevados. YOLOv3 presenta también muy buenos resultados ya que sus valores son próximos a los de *RetinaNet*. Se observa que la precisión media tomando como referencia una intersección de 0.5 *IoU,* los valores de los tres algoritmos no presentan valores muy distantes. Pero esto cambia si se incrementa el valor de intersección a 0.75, en el que YOLOv2 presenta muy baja precisión. El aumento de precisión de YOLOv3 a diferencia de la versión anterior, se debe al conjunto de mejoras que se han incorporado en esta versión. Además, se observa que YOLOv3 presenta mejor precisión que SSD.

En la siguiente tabla se observan las principales diferencias entre las tres versiones de YOLO y en anexos se muestra un estudio en el que se observan las diferencias entre las versiones en función de gráficas Precisión-*Recall* de los resultados obtenidos. Las características han sido obtenidas en base a lo desarrollado previamente y los datos de la velocidad alcanzada con Titan X se han tomado de (You Only Look Once: Unified, Real-Time Object Detection, 2016) [97] y (Towards Data Science, 2018) [95].

	YOLOv1	YOLOv2	YOLOv3
Nº de capas	26	30	106
Nº de *boxes bounding*	2	5	3
Modo de clasificación	Error cuadrático	Softmax	Pérdida de entropía
Característica	Sus dos últimas capas son lineales	Entrenamiento dividido en dos fases	Factor *stride* de valores 32, 16 y 8
Detección de objetos pequeños	Precisión baja	Precisión media	Precisión alta
Velocidad	Velocidad intermedia	Más rápido que el resto	Más lento que los otros
Velocidad alcanzada con Titan X	45 FPS	45 o 90 FPS	30 FPS

Tabla 26: Diferencias entre las versiones de YOLO

5.5.4 Experimentación YOLOv3

El algoritmo YOLO tiene tres modos de ejecución uno de ellos es realizar la detección de los objetos mediante imágenes o fotografías, el otro modo es mediante un vídeo y el tercer modo es mediante la cámara, que en este caso se utilizó la incorporada en la tarjeta Jetson TX2. YOLOv3 presenta una red neuronal preentrenada denominada COCO que detecta hasta 80 clases de objetos. Las pruebas que se muestran a continuación se realizaron con la SBC Jetson TX2 y se utilizó la red neuronal preentrenada COCO. Se realizaron las pruebas de los tres modos posibles, alcanzado 3 FPS. Posteriormente se entrenó una nueva red con datos propios para que detecte nuevas clases o tipos de objetos.

YOLOv3 imágenes

Se realizaron diversas pruebas para las cuales se descargaron diferentes imágenes, para comprobar de este modo la capacidad de detección de YOLOv3. En las imágenes se observa que la detección que realiza este algoritmo es muy buena, ya que es capaz de detectar varios objetos en una misma imagen, como es el caso de la primera y tercera fotografía.

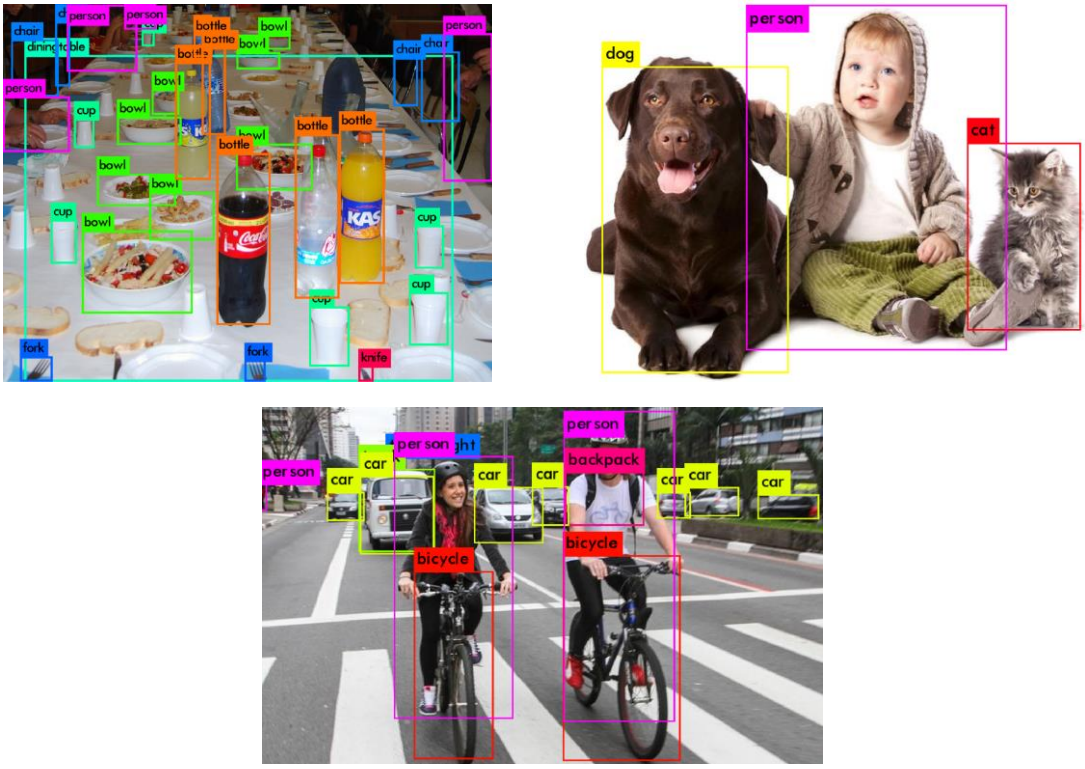


Figura 143: YOLOv3 Imágenes

YOLOv3 vídeos

Para realizar estas pruebas fue necesario descargar los vídeos en formato .mp4. Las imágenes siguientes fueron capturadas de dos de estos vídeos en el que se observa que los resultados obtenidos son muy buenos. Al igual que en el caso anterior es capaz de detectar varios objetos en una misma imagen, pero en este caso en movimiento.

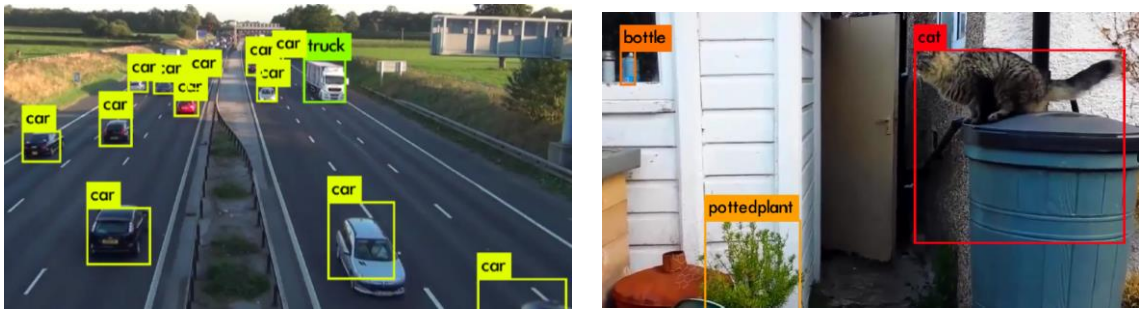


Figura 144: YOLOv3 vídeos

YOLOv3 cámara incorporada

Para realizar estas pruebas se utilizaron diferentes clases de objetos, de los cuales dos imágenes se muestran a continuación. Con la primera imagen se pretende demostrar lo explicado anteriormente sobre la clasificación de la etiqueta. Se observa que hay un peluche entre el coche rosa y el reloj, pero este además de ser peluche tiene forma de perro, por lo que puede ser detectado de las dos formas. Durante las pruebas se observó que lo detectaba en algunas ocasiones como peluche y en otras como perro. Esto se debe al valor de probabilidad que presenta con referencia a que pertenezca a esa clase. Por lo que la asignación de la etiqueta dependerá de factores como el enfoque de la cámara, iluminación, distancia, ya que todo esto influye a la hora de realizar la predicción. Por ejemplo, si este objeto se encontraba muy cerca de la cámara era detectado como peluche, sin embargo, colocado a una cierta distancia lo detectaba como perro.

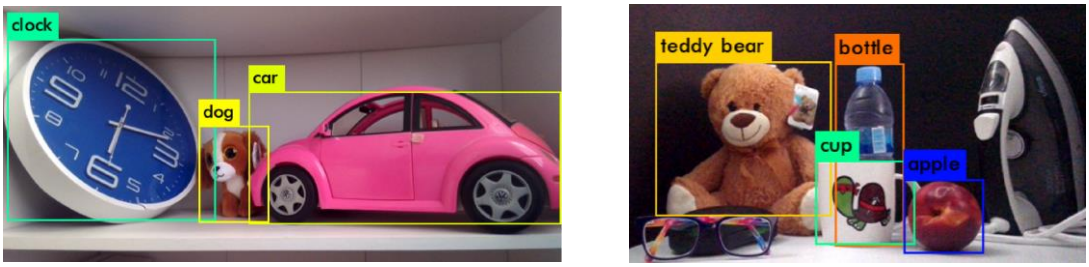


Figura 145: YOLOv3 Cámara

En la segunda imagen se observa que dos objetos no fueron detectados, esto se debe a que estas clases no se encuentran en la lista de las clases de objetos de la red neuronal preentrenada COCO, esta lista de las clases se encuentra en anexos. El resto de los objetos fueron detectados satisfactoriamente comprobando así que YOLOv3 es un algoritmo muy eficiente y con una precisión muy buena.

Capítulo 6

Reentrenamiento de redes neuronales

6.1.Entrenamiento YOLOv3

El nuevo entrenamiento de la red neuronal que se realizó fue en base al algoritmo YOLOv3, aunque se puede realizar con otros diversos algoritmos como ImageNet. Se realizaron un par de entrenamientos, de los cuales tres se mostrarán a continuación. Con estos tres entrenamientos se pretende mostrar los diferentes resultados que se obtuvieron cambiando el valor de algunos parámetros.

- Primer entrenamiento: se realizó para que detecte botellas de Coca-Cola y Fanta de naranja.
- Segundo entrenamiento: ha de detectar las latas de estos dos refrescos.
- Tercer entrenamiento: en este caso ha de detectar cualquier logo de estas dos marcas de bebida en cualquier objeto.

A continuación, se describe de forma general los pasos que se realizaron para llevar a cabo el entrenamiento, y posteriormente en cada uno de los tres ejemplos se explicará los cambios que se realizaron y los resultados que se obtuvieron.

Para realizar el entrenamiento de la red se siguió el método utilizado por (Medium, 2018) [54] y (Redmon, 2018) [83] el cual se inicia descargando las imágenes que se van a utilizar para el entrenamiento en formato .jpg. Es recomendable que este conjunto de imágenes este formado tanto por imágenes perfectas en la que sea muy fácil el reconocimiento del objeto y por otras que hagan que el aprendizaje resulte más difícil. Una vez descargadas las imágenes se marcan los cuadros que limitan los objetos que se deben aprender y detectar posteriormente.

Figura 146: Cuadro límite del objeto real

Cada imagen genera un archivo .txt en el que aparece las coordenadas de la etiqueta del cuadro marcado, que corresponden a las que aparecen en el cuadrado de la derecha. En este cuadrado aparecen todas las coordenadas de los respectivos cuadros marcados, ya que se puede marcar a más de un objeto.

El formato que presenta este archivo .txt se ha de cambiar para que YOLOv3 lo procese. Los valores que se muestran en este archivo son los siguientes:

[Cuadro delimitador izquierda X] [Cuadro delimitador superior Y] [Cuadro delimitador derecha X] [Cuadro delimitador inferior Y]

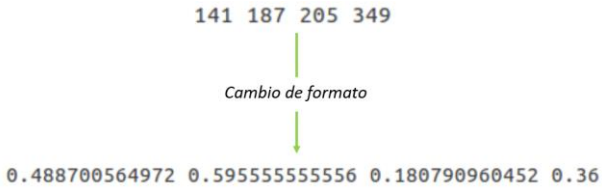


Figura 147: Cambio de formato

Estos valores son cambiados de formato, y al cambiarlos con su respectivo comando se generan nuevos archivos .txt, en cuyo interior aparecen sus respectivos valores que indican:

[Centro del objeto en X] [Centro del objeto en Y] [Ancho del objeto en X] [Ancho del objeto en Y]

Posteriormente las imágenes que habían sido descargadas en formato .jpg se convierten a .JPEG, mediante el comando:

*mogrify -format JPEG *.jpg*

Ahora se generan los archivos *train.txt* y *test.txt*, los cuales muestran las imágenes que va a utilizar la red neuronal para el entrenamiento, que en este caso son el 50 % del total de las imágenes.

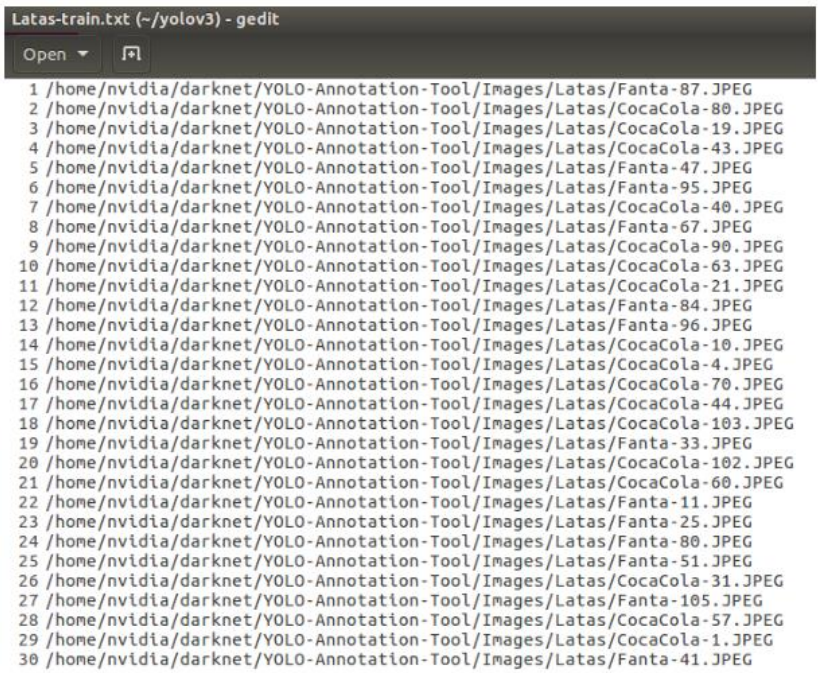


Figura 148: Archivo train.txt

Latas-test.txt (~/yolov3) - gedit

Open

```
 1 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-39.JPEG
 2 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-63.JPEG
 3 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-56.JPEG
 4 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/CocaCola-97.JPEG
 5 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/CocaCola-48.JPEG
 6 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/CocaCola-81.JPEG
 7 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/CocaCola-23.JPEG
 8 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-42.JPEG
 9 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/CocaCola-59.JPEG
10 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-70.JPEG
11 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-52.JPEG
12 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/CocaCola-22.JPEG
13 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-40.JPEG
14 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-49.JPEG
15 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-88.JPEG
16 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-74.JPEG
17 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-97.JPEG
18 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/CocaCola-17.JPEG
19 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-78.JPEG
20 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/CocaCola-46.JPEG
21 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/CocaCola-49.JPEG
22 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/CocaCola-47.JPEG
23 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-28.JPEG
24 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-68.JPEG
25 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-89.JPEG
26 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/CocaCola-64.JPEG
27 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-92.JPEG
28 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/Fanta-31.JPEG
29 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/CocaCola-15.JPEG
30 /home/nvidia/darknet/YOLO-Annotation-Tool/Images/Latas/CocaCola-51.JPEG
```

Figura 149: Archivo test.txt

Después se crean dos archivos a los cuales se le pone de extensión *.data* y *.names*. En el primero se indican datos como el número de clases que se va a entrenar o los nombres de los archivos train.txt y test.txt. Mientras que en el segundo se señala los nombres de las clases que la red ha de aprender.

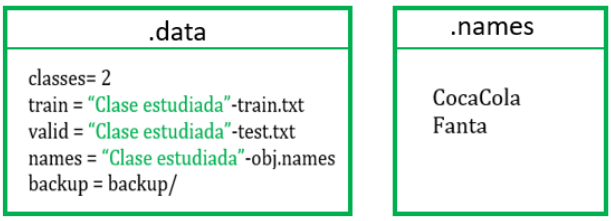


Figura 150: Archivos .data y .names

El entrenamiento se puede realizar mediante *tiny-yolo.cfg* y *yolov3.cfg*. La primera se utiliza en el caso de que la GPU sea menor de 2 GB, mientras que el segundo modo se utiliza para GPUs de más de 4 GB. Si se tiene la posibilidad de elegir, es preferible utilizar *yolov3.cfg* ya que es más preciso que el otro. En este caso como es posible realizarlo con *yolov3.cfg*, ya que la GPU de la tarjeta Jetson TX2 comparte 8GB con la RAM, se pudo realizar con este.

El paso previo a iniciar el entrenamiento es modificar si fuera necesario el archivo *yolov3.cfg,* en él se indican parámetros como la resolución de la imagen, la activación del *Random*, entre otros. Este se mostrará con mayor detalle en cada uno de los entrenamientos. Es recomendable aumentar la velocidad del reloj de muestreo, aumentando así su rendimiento. Esto se realiza mediante el comando:

sudo nvpmodel -m 0

Se ha de tener en cuenta que durante el entrenamiento se generan archivos en los que se guardan las ponderaciones realizadas. Estas ponderaciones se guardan cada 100 iteraciones hasta llegar a los 900, y después cada 10000 iteraciones. En el caso de que no se quiera esperar a que realice tantas iteraciones, este valor de 10000 se puede cambiar y comprobar si el entrenamiento ha sido satisfactorio. En este caso este dato se modificó y se cambió en el archivo *datector.c* para que se guardara cada 2000 ponderaciones. Por lo que al llegar a 900, el siguiente archivo que se generaría sería el de 2000, el siguiente 4000 y así sucesivamente. Finalmente se pone a entrenar a la red neuronal, ejecutando el comando:

./darknet detector tren cfg/Latas-obj.data cfg/Latas-yolov3.cfg darknet53.conv.74

Cuando la red neuronal está realizando el entrenamiento, este se puede detener y realizar comprobaciones con los archivos generados para comprobar si detecta correctamente. En el caso de realizar las pruebas y comprobar que la red aún no ha entrenado lo suficiente se puede retomar el entrenamiento. Esto se realiza ejecutando el comando:

./darknet detector train cfg/Latas-obj.data cfg/Latas-yolov3.cfg backup/yolov3.backup

La siguiente imagen se señalan los datos más relevantes que aparecen por el terminal durante el entrenamiento.

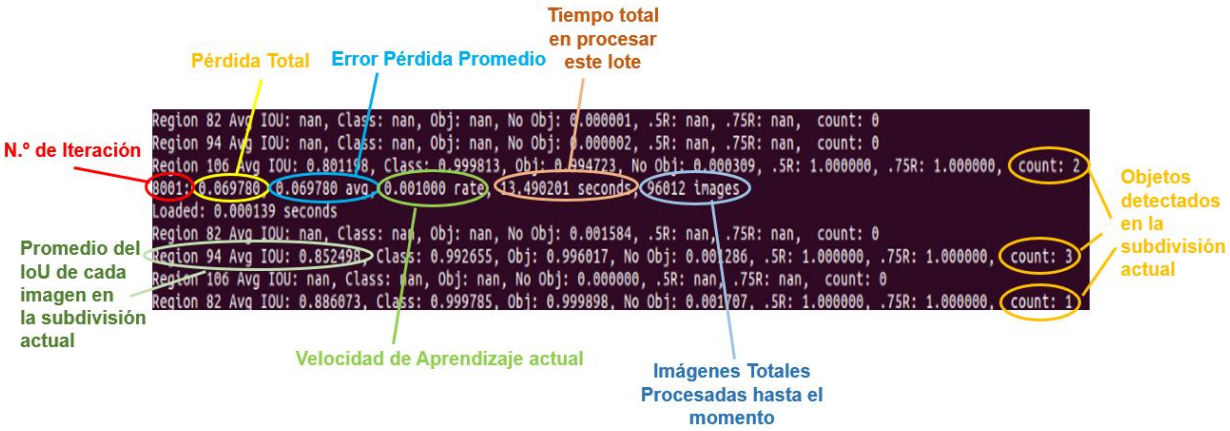


Figura 151: Datos generados durante el entrenamiento

Una aproximación para realizar las primeras pruebas es que se superen las 2000 iteraciones realizadas. Otro dato a tener en cuenta es el error de pérdida promedio (avg), el cual cuanto más pequeño sea, mejores resultados se obtendrán. Se ha de tener en cuenta que los resultados obtenidos cuando avg está por debajo de 0.06 no varían mucho, aunque siga entrenando la red. Ya que por debajo este valor el aprendizaje de la red es imperceptible. (Timebutt.io, 2017) [93]

6.1.1 Entrenamiento de botellas

Para este entrenamiento se utilizaron 150 imágenes de cada clase y con el cuadro que se limita a las botellas se pretendía coger toda la botella completa o lo máximo posible, como se muestra en las siguientes imágenes.

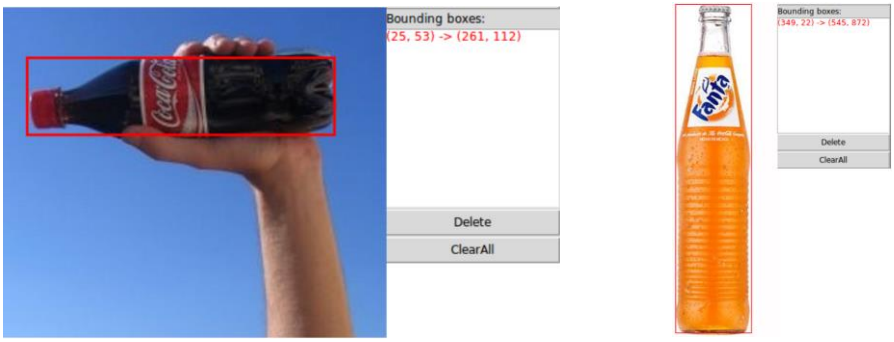


Figura 152: Captura de la botella

Se utilizaron 150 imágenes en total para realizar el entrenamiento que fueron las que tomaron aleatoriamente los archivos *train.txt* y *test.txt*. En este caso el archivo *yolov3.cfg* se modificó con los siguientes valores:

- Batch: 24, por lo que se utilizarán 24 imágenes por iteración.
- Subdivisión: 24, este valor divide el lote.
- Filtros: 21, este dato se obtiene en función del número de clases. Cuya expresión es $filtros = (clases + 5) \cdot 3$
- Clases: 2, ya que sólo se está entrenando para que detecte las botellas de los refrescos Coca-Cola y Fanta.
- Random: 1, por lo que está activado y cada diez lotes reducirá la imagen de manera aleatoria.
- Anchura: 416.
- Altura: 416.

Dos de los datos que se pueden modificar es la anchura y altura de las imágenes de entrada. Por lo que se recomiendan que las imágenes tengan un tamaño parecido, ya que si son muy pequeñas en la entrada estas imágenes se ampliaran hasta obtener la resolución de (416x416). Además, al tener las imágenes del mismo tamaño facilitaría el procesamiento paralelo de la GPU, evitando de este modo problemas de concatenación al implementar diferentes algoritmos.

Otro parámetro para tener en cuenta es el batch y subdivisión, ya que estos valores indican cuantas imágenes por lotes van a ser utilizadas. En un principio se le asigno el valor de 64 al batch y a la subdivisión de 8, para obtener de este modo ocho lotes de ocho imágenes cada una. Pero al poco tiempo el sistema se quedó sin memoria por lo que estos valores se modificaron a 64 y 32 respectivamente, pero se obtuvo el mismo problema. Se realizó un tercer intento en el que el valor del bach era 24 y la subdivisión 12, pero se obtuvo de igual modo el mismo problema. Finalmente asignándoles el valor de 24 a ambos funcionó, generando de este modo 24 lotes en cada iteración, con una imagen respectivamente.

El entrenamiento realizó 2000 iteraciones y tardó aproximadamente 18 horas, obteniendo un valor de avg igual a 0.1753.

Con estos valores se realizaron las pruebas del entrenamiento obteniendo resultados no muy buenos, ya que no siempre detectaba las botellas. En algunas ocasiones detectaba la botella e incluso a ambos tipos de botellas como se muestra en las imágenes inferiores. Ambas imágenes fueron capturadas por la cámara de la tarjeta Jetson TX2.

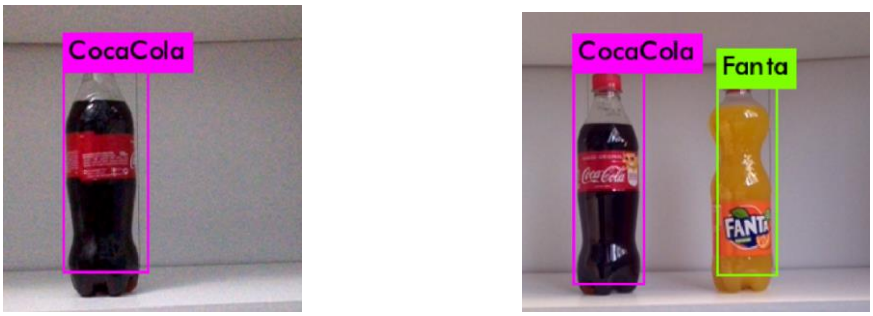


Figura 153: Detección de botellas

En otras pruebas sólo identificaba una de las dos o no detectaba a ninguna, se ha de tener en cuenta que la detección la realiza en función del color del líquido, por lo que si la botella está vacía no la detectará en ningún momento.

Figura 154: Resultado de las botellas

Que los resultados obtenidos no sean muy buenos se debe a que sólo se realizaron 2000 iteraciones y el valor de avg era aún elevado.

6.1.2 Entrenamiento de latas

Para realizar este entrenamiento se utilizaron 110 imágenes de cada tipo de latas, por lo que los archivos *test.txt* y *train.txt* tomaron 110 imágenes para el entrenamiento, que son la mitad de las imágenes totales. En este caso la zona de la imagen limitada por el cuadro se ajustó lo máximo posible al interior de las latas como se muestra en las siguientes imágenes.

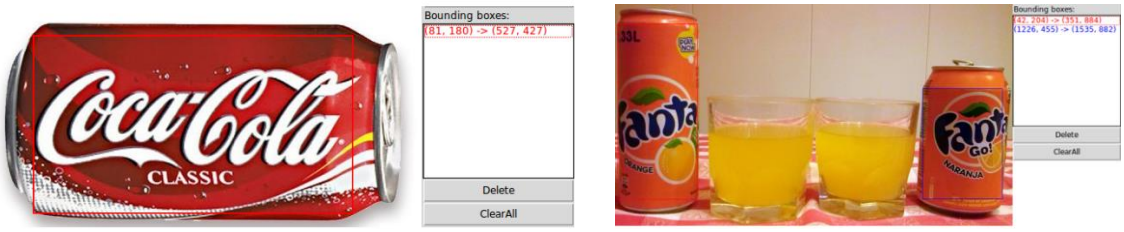


Figura 155: Detección de latas

Para comprobar como los valores del batch y división influían en el entrenamiento se modificaron y en este caso el archivo *yolov3.cfg* presentaba los siguientes valores:

- Batch: 12, por lo que se utilizarán 12 imágenes por cada iteración.
- Subdivisión: 12.
- Filtros: 21.
- Clases: 2, por lo que detectará las latas de las bebidas Coca-Cola y Fanta.
- Random: 1, por lo que está activado y cada diez lotes reducirá la imagen de manera aleatoria.
- Anchura: 416.
- Altura: 416.

El entrenamiento tardó aproximadamente 29 horas en realizar 6000 iteraciones y en este caso los resultados obtenidos fueron algo mejor que los anteriores. Esta mejora se debe a que se realizaron más iteraciones y el valor obtenido avg era de 0.0714, este dato es lo suficientemente pequeño por lo que se puede dar cómo bueno.

En este caso se generan 12 lotes con una imagen por iteración, por lo que si se compara con el entrenamiento de las botellas se observa que realiza más iteraciones en menos tiempo. Esto se debe a que en cada iteración utiliza menos lotes de imágenes, por lo que en el caso anterior utilizaba 24 y en este 12.

Durante el entrenamiento se tomaron nota de los valores que se estaban obteniendo dos de estos datos es que con 2786 iteraciones realizadas se había obtenido un avg de 0.219 y con 5776 iteraciones el valor era de 0.0720. Con el valor de la primera iteración se observa que con un menor número de iteraciones se obtuvo un mejor valor de avg.

Con esto se deduce que, a cambio de incrementar la velocidad de las iteraciones, se obtiene un peor valor del error de pérdida promedio.

Realizando las pruebas se comprobó que la mayoría de las veces las latas eran detectadas, pero no siempre, ya que en alguna prueba se observó que no eran del todo identificadas.

Figura 156: Resultado de las latas

Se cree que los resultados hubieran sido mucho mejores si el cuadro delimitador de la lata la hubiera enmarcado por completo. Esta suposición se plantea en base a los resultados obtenidos de las pruebas realizadas con la cámara y fotos.

Los resultados de las fotos eran muy buenos, en la que se detectaba en la mayoría de las ocasiones las latas, presentando sólo un pequeño error, debido a las latas no detectadas. Sin embargo, a la hora de realizar las pruebas con la cámara, para que la lata sea detectada dependía de la distancia, la luz o que estuviera bien colocada. Esto se debe a que la red neuronal ha realizado el aprendizaje en base los colores de las latas y no de la forma, por lo que factores como la luz, el brillo o la distancia puede afectar en su detección.

6.1.3 Entrenamiento de logos

Por último, se realizó el entrenamiento con los logos de las marcas Coca-Cola y Fanta, de este modo se pretende que sea capaz de detectar cualquier objeto que tenga estos logos. Para ello se utilizaron 160 imágenes de cada logo y en estas aparecían diferentes objetos como latas, botellas, tapas, camisetas o frisbee, como se muestran a continuación.

Figura 157: Detección de logos

En este caso los valores del batch y subdivisón se mantuvieron por lo que el archivo *yolov3.cfg* no se modificó:

- Batch: 12, por lo que se utilizará 1 imagen por cada lote.
- Subdivisión: 12.
- Filtros: 21.
- Clases: 2, por lo que detectará cualquier objeto que contenga el logo de Coca-Cola y Fanta.
- Random: 1, por lo que está activado y cada diez lotes reducirá la imagen de manera aleatoria.
- Anchura: 416.
- Altura: 416.

El entrenamiento duró 36 horas y se realizaron 8000 iteraciones, obteniendo un valor de avg igual a 0.0697. Este valor es lo más pequeño posible por lo que se considera como ideal, ya que, aunque siga entrenando la red neuronal el aprendizaje será pobre, por lo que no vale la pena seguirlo entrenando.

En las siguientes imágenes se observan que los logos de ambos refrescos fueron detectados en latas, botellas y otros objetos, como una pala de playa. En la segunda imagen se observa que, aunque la botella este vacía sí que la detecta ya que en este caso lo que se detecta es el logo de las marcas de las bebidas.

Figura 158: Resultados de logos

En este caso los resultados que se obtuvieron fueron mejores, pero no del todo correcto, ya que se observó que la red había sido sobreentrenada. Esto se debe a que el modelo utilizado de la red es complejo en relación con los datos de entrada, que en este caso sólo tenía que detectar dos clases diferentes. Por lo que el entrenamiento realizado se ajusta tanto a los datos de entrada, que no ha sido capaz de entrenar como para generalizar y relacionar nuevos ejemplos. Con esto se determina que la red en los tres casos fue sobreentrenada, y que en los otros dos casos anteriores factores como un valor de avg elevado y el modo de limitar el cuadro para que la red entrene, influyeron aún más en los resultados obtenidos.

Capítulo 7

Conclusiones

La gran potencia y prestaciones que ofrece el módulo Jetson TX2 en un tamaño tan reducido hacen que sea el sistema más apropiado para el desarrollo de aplicaciones de visión artificial y aprendizaje profundo, en las que el consumo de potencia y el tamaño sea importante. Esto es gracias a que lleva incorporado un controlador de memoria quien controla el acceso de dispositivos a esta, aumentando así su rendimiento y minimizando el consumo de energía. Por lo que la potencia que el módulo Jetson TX2 requiere para su funcionamiento es de sólo 15 W.

La posibilidad de realizar la activación del modo de protección mediante hardware o software hacen que el sistema este controlado térmicamente de manera permanente. Y al llevar incorporado un disipador de calor se consigue que la temperatura se distribuya de manera uniforme evitando de este modo el sobrecalentamiento de determinadas zonas. Aunque lo que realmente la convierte en un dispositivo de gran interés es la arquitectura Pascal que presenta su GPU. Esta al ser compatible con la memoria HBM2 realiza el procesamiento de datos a mayor ancho de banda, mejorando de este modo su alta eficiencia. Además, al incorporar la memoria LPDDR4, los núcleos Denver y ARM-A57, de la CPU y los núcleos CUDA de la GPU hace que sea uno de los dispositivos idóneos para incorporar en aplicaciones donde la velocidad del procesamiento de datos sea importante. Al contar con sistemas de protección como TSEC o ECC hacen que sea un sistema seguro con los que ante un fallo el sistema sea capaz de recuperase. Otra de las ventajas que presenta es que la frecuencia de reloj de muestreo de la GPU y CPU se pueden aumentar, y además se puede llevar a cabo la activación o desactivación de sus respectivos núcleos. Esto se comprobó ya que durante el reentrenamiento de la red neuronal se aumentó la frecuencia de la CPU a 2 GHz y a la GPU a 1.3 GHz.

Su conexión Wifi y bluetooth evitan que se requiera de cables para su conexión a la red, por lo que no se depende en un principio del lugar de trabajo, siempre que este tenga buena conexión. Al contar con conexión USB 3.0 de 5GB/s hace que la transferencia de datos se realice a una velocidad muy superior y mediante su conexión HDMI genera imágenes de mejor resolución, haciendo todo esto que la interacción con periféricos sea de mejor calidad. Además, se le puede incorporar al sistema más dispositivo como por ejemplo cámaras o discos duros, mediante conectores como SATA o PCIe cuya velocidad alcanza 5 GT/s.

Al realizar la instalación del SO Linux en el equipo no se obtuvo ningún problema, ya que es compatible su estancia en un equipo con dos SO. A medida que se iban instalando los algoritmos o librerías necesarias se comprobó que no es necesario reiniciar siempre el sistema tras una actualización o instalación, y además permite la opción de actualizar sólo la parte que se requiere. Al ejecutar los ejemplos de CUDA 8.0 se comprobó que es compatible con varias versiones de OpenGL, ya que en los ejemplos ejecutados utilizaban diferentes versiones. Otra de las ventajas que presenta, aunque no se llegó a probar es que los parámetros de los shaders pueden ser modificados. Todo esto hace que el procesamiento de imágenes, el aprendizaje

profundo o el análisis de gráficos se realicen mucho más rápido. Al contar con la herramienta TensorRT se obtiene un mayor rendimiento en el entrenamiento de las redes neuronales y permite además implementar nuevas capas personalizadas, aunque esto último no se llegó a comprobar. Otra de las herramientas que aceleran este entrenamiento de estas redes es CuDNN quien, además al presentar una estructura *pipeline* evita el cuello de botella, ya que el procesamiento no se realiza en serie sino en paralelo.

Al llevar a cabo la experimentación de los distintos ejemplos se ha comprobado de la gran eficiencia y rendimiento que ofrece la SBC TX2. Desde la ejecución de Jetson-Inference donde se obtuvieron muy buenos resultados, hasta los ejemplos de CUDA 8.0 que, aunque son sencillos se ha observado el funcionamiento de OpenGL. Mediante Jetson-Reinforcement se comprendió mejor en que consiste que un sistema sea entrenado en función de su entorno. Estos ejemplos presentan un entrenamiento similar al no supervisado y es en esto en lo que muchas empresas están trabajando. Además, actualmente este modo de entrenamiento se está utilizando en muchos sistemas, en los que se entrenan de manera independiente y posteriormente son incorporados por ejemplo a robots.

Por último, el reentrenamiento de la red neuronal como ya se ha dicho anteriormente no fue del todo idóneo, ya que presento un sobreentrenamiento. Aun así, sí que llegó a detectar las diferentes marcas y hacer una predicción aceptable, sin embargo, la eficiencia de la red preentrenada que presenta YOLOv3 es muy buena, en la que se obtuvo prácticamente entorno al 95% de aciertos.

Los ejemplos mostrados en este TFG sólo han sido una pequeña muestra de un conjunto de algoritmos y aplicaciones que se pueden llegar a realizar con este sistema. Como se dice “esto sólo es la punta del iceberg” y aún quedan por conocer aplicaciones que se están desarrollando o quedan por desarrollarse en un futuro en base a Jetson TX2.

Capítulo 8

Normas y referencias

8.1.Programas utilizados

Para el funcionamiento de la tarjeta Jetson TX2 y la realización de las pruebas se descargaron diferentes programas o algoritmos. Estos se descargaron de la plataforma Github (GitHub, 2018) y se implementaron en la SBC. Los programas utilizados son los que se muestran a continuación.

CUDA 8.0:

- Partículas
- Fluido
- Random Fog

Jetson-Inference:

- ImageNet
- DetectNet
- SegNet

Jetson-reinforcement:

- Cartpole
- Lunar Lander
- Brazo Robótico

OpenCV

- Conversión RGB a gris
- Suavizado
- Detector Canny

YOLOv3

8.2.Bibliografía

8.2.1 Búsqueda de información

[1]. Acodigo.blogspot.com. (2018). Introducción a la Programación GLSL. [online] Disponible en: http://acodigo.blogspot.com/2016/09/introduccion-la-programacion-glsl.html [Accedido el 17 Aug. 2018].

[2]. Acodigo.blogspot.com. (2018). OpenCV. [online] Disponible en: http://acodigo.blogspot.com/p/tutorial-opencv.html [Accedido el 15 Aug. 2018].

[3]. Advances in Cryptology-ASIACRYPT 2007. (2007). .

[4]. Álvarez, R. (2018). Aquí está la primera demo del dispositivo de Magic Leap... y no se parece en nada a lo que nos han querido vender desde hace años. [online] Xataka.com. Disponible en: https://www.xataka.com/realidad-virtual-aumentada/aqui-esta-primera-demo-dispositivo-magic-leap-no-se-parece-nada-que-nos-han-vendido-hace-anos [Accedido el 11 Aug. 2018].

[5]. Anon, (2018). [online] Disponible en: https://developer.nvidia.com/embedded/jetpac [Accedido el 29 Jul. 2018].

[6]. Arrabales, R. (2018). Deep Learning: qué es y por qué va a ser una tecnología clave en el futuro de la inteligencia artificial. [online] Xataka.com. Disponible en: https://www.xataka.com/robotica-e-ia/deep-learning-que-es-y-por-que-va-a-ser-una-tecnologia-clave-en-el-futuro-de-la-inteligencia-artificial [Accedido el 29 Aug. 2018].

[7]. Blog.echen.me. (2018). Exploring LSTMs. [online] Disponible en: http://blog.echen.me/2017/05/30/exploring-lstms/ [Accedido el 2 Sep. 2018].

[8]. Caparrini, F. and Work, W. (2018). Clasificación Supervisada y No Supervisada - Fernando Sancho Caparrini. [online] Cs.us.es. Disponible en: http://www.cs.us.es/~fsancho/?e=77 [Accedido el 2 Sep. 2018].

[9]. Debian.org. (2018). Debian -- El sistema operativo universal. [online] Disponible en: https://www.debian.org/index.es.html [Accedido el 13 Mar. 2018].

[10]. DCV-01672-N2-GP, C. (2018). Coolermaster Heatsink and Fan - DCV-01672-N2-GP. [online] Silicon Highway - Enabling Technologies, NVIDIA JETSON Developments kits, TK1, TX1, TX2, Connect Tech. Disponible en: https://www.siliconhighwaydirect.co.uk/Cooler-Master-DCV-01672-N2-GP-p/dcv-01672-n2-gp.htm [Accedido el 20 Jul. 2018].

[11]. Devtalk.nvidia.com. (2017). [online] Disponible en: https://devtalk.nvidia.com/default/topic/1027789/jetson-tx2/jetsonhacks-car-video-detection-example-needs-update/ [Accedido el 20 Apr. 2018].

[12]. Devtalk.nvidia.com. (2017). [online] Disponible en: https://devtalk.nvidia.com/default/topic/1003688/jetson-tx1/-the-image-from-tx1-onboard-vision-sensor-is-upside-down-/ [Accedido el 25 Jul. 2018].

[13]. Docs.nvidia.com. (n.d.). cuDNN Developer Guide :: Deep Learning SDK Documentation. [online] Disponible en: https://docs.nvidia.com/deeplearning/sdk/cudnn-developer-guide/index.html [Accedido el 15 Apr. 2018].

[14]. Documentación informática industrial y comunicaciones. (2018)..

[15]. Elinux.org. (2018). Jetson TX2 - eLinux.org. [online] Disponible en: https://elinux.org/Jetson_TX2 [Accedido el 10 Feb. 2018].

[16]. Es.wikipedia.org. (2018). Canonical. [online] Disponible en: https://es.wikipedia.org/wiki/Canonical [Accedido el 2 Sep. 2018].

[17]. Es.wikipedia.org. (2018). Direct3D. [online] Disponible en: https://es.wikipedia.org/wiki/Direct3D [Accedido el 2 Sep. 2018].

[18]. Es.wikipedia.org. (2018). GDDR3. [online] Disponible en: https://es.wikipedia.org/wiki/GDDR3 [Accedido el 2 Sep. 2018].

[19]. Es.wikipedia.org. (2018). GDDR5. [online] Disponible en: https://es.wikipedia.org/wiki/GDDR5 [Accedido el 3 Sep. 2018].

[20]. En.wikipedia.org. (2018). High Bandwidth Memory. [online] Disponible en: https://en.wikipedia.org/wiki/High_Bandwidth_Memory [Accedido el 3 Sep. 2018].

[21]. En.wikipedia.org. (2018). Long short-term memory. [online] Disponible en: https://en.wikipedia.org/wiki/Long_short-term_memory [Accedido el 2 Sep. 2018].

[22]. En.wikipedia.org. (2018). Nvidia Tesla. [online] Disponible en: https://en.wikipedia.org/wiki/Nvidia_Tesla [Accedido el 31 Aug. 2018].

[23]. Es.wikipedia.org. (2018). OpenGL. [online] Disponible en: https://es.wikipedia.org/wiki/OpenGL#OpenGL_2.1 [Accedido el 1 Sep. 2018].

[24]. Es.wikipedia.org. (n.d.). Shaders Unificados. [online] Disponible en: https://es.wikipedia.org/wiki/Shaders_Unificados [Accedido el 1 Sep. 2018].

[25]. En.wikipedia.org. (2018). Tesla (microarchitecture). [online] Disponible en: https://en.wikipedia.org/wiki/Tesla_(microarchitecture) [Accedido el 1 Sep. 2018].

[26]. Es.wikipedia.org. (2018). Unidad de procesamiento tensorial. [online] Disponible en: https://es.wikipedia.org/wiki/Unidad_de_procesamiento_tensorial [Accedido el 2 Sep. 2018].

[27]. Evaluating State-of-the-art Object Detector on Challenging Traffic Light Data. (n.d.)..

[28]. FDDB: A Benchmark for Face Detection in Unconstrained Settings. (n.d.). .

[29]. for?, W. (2013). What are shaders in openGL and what do we need them for?. [online] Stack Overflow. Disponible en: https://stackoverflow.com/questions/17789575/what-are-shaders-in-opengl-and-what-do-we-need-them-for [Accedido el 1 Sep. 2018].

[30]. Franklin, D., Franklin, D., Franklin, D., Franklin, D., Gray, A., Gottbrath, C., Olson, R. and Prasanna, S. (2018). JetPack 3.1 Doubles Jetson's Low-Latency Inference Performance | NVIDIA Developer Blog. [online] NVIDIA Developer Blog. Available at: https://devblogs.nvidia.com/jetpack-doubles-jetson-inference-perf/ [Accedido el 29 Jul. 2018].

[31]. Genie.webierta.skn1.com. (2018). wiki:gstreamer [Genie Doc]. [online] Disponible en: http://genie.webierta.skn1.com/wiki/gstreamer [Accedido el 3 Aug. 2018].

[32]. Gstreamer.freedesktop.org. (2018). Release notes forGStreamer 1.8.2"". [online] Disponible en: https://gstreamer.freedesktop.org/releases/gstreamer/1.8.2.html [Accedido el 1 Aug. 2018].

[33]. GitHub. (n.d.). dusty-nv/jetson-inference. [online] Disponible en: https://github.com/dusty-nv/jetson-inference#using-the-console-program-on-jetson [Accedido el 25 Jul. 2018].

[34]. GitHub. (n.d.). NVIDIA/DIGITS. [online] Disponible en: https://github.com/NVIDIA/DIGITS [Accedido el Aug. 2018].

[35]. Harris, M., Harris, M., Kelmelis, E., Sakharnykh, N. and Luitjens, J. (2018). Maxwell: The Most Advanced CUDA GPU Ever Made. [online] NVIDIA Developer Blog. Disponible en: https://devblogs.nvidia.com/maxwell-most-advanced-cuda-gpu-ever-made/ [Accedido el 1 Sep. 2018].

[36]. Hyper-Q Example. (2013). .

[37]. IGN - Soluciones de gestión para pymes. (2018). Historia del Big Data: un largo viaje poco conocido.. [online] Disponible en: https://ignsl.es/historia-del-big-data/ [Accedido el 31 Aug. 2018].

[38]. INA3221 Triple-Channel, High-Side Measurement, Shunt and Bus Voltage Monitor with I 2C- and SMBUS-Compatible Interface. (2016). .

[39]. INTRODUCCIÓN A LA PROGRAMACIÓN EN CUDA. (2016)..

[40]. JetsonHacks. (2017). NVPModel - NVIDIA Jetson TX2 Development Kit - JetsonHacks. [online] Disponible en: https://www.jetsonhacks.com/2017/03/25/nvpmodel-nvidia-jetson-tx2-development-kit/ [Accedido el 15 Aug. 2018].

[41]. Jedec.org. (2015). e.MMC | JEDEC. [online] Disponible en: https://www.jedec.org/standards-documents/technology-focus-areas/flash-memory-ssds-ufs-emmc/e-mmc [Accedido el 10 Dec. 2017].

[42]. Keras.io. (n.d.). Keras Documentation. [online] Disponible en: https://keras.io/ [Accedido el 5 Aug. 2018].

[43]. Keras.io. (n.d.). Guide to the Functional API - Keras Documentation. [online] Disponible en: https://keras.io/getting-started/functional-api-guide/ [Accedido el 5 Aug. 2018].

[44]. Kingston Technology Company. (n.d.). eMMC, con capacidades de hasta 64 GB | Kingston. [online] Disponible en: https://www.kingston.com/es/embedded/emmc [Accedido el 11 Dec. 2017].

[45]. labels?, I. (2018). Is it okay to use cross entropy loss function with soft labels?. [online] Cross Validated. Disponible en: https://stats.stackexchange.com/questions/206925/is-it-okay-to-use-cross-entropy-loss-function-with-soft-labels [Accedido 28 Aug. 2018].

[46]. Lang, B. (2018). Magic Leap Reveals Developer Demo, Confirms NVIDIA TX2 Hardware. [online] Road to VR. Disponible en: https://www.roadtovr.com/magic-leap-ar-developer-demo-nvidia-tx2-cpu-gpu-hardware/ [Accedido el 11 Aug. 2018].

[47]. Large Scale Distributed Deep Networks. (n.d.). .

[48]. Lavidaconubuntu.blogspot.com.es. (2018). La Vida con Ubuntu. [online] Disponible en: http://lavidaconubuntu.blogspot.com.es/ [Accedido el 25 Jul. 2018].

[49]. Learnopengl.com. (2014). LearnOpenGL - Shaders. [online] Disponible en: https://learnopengl.com/Getting-started/Shaders [Accedido el 1 Sep. 2018].

[50]. LG (2016). ¿Qué es la Memoria Caché, y Para Qué Sirve? | LG Blog. [online] LG Chile Blog Oficial. Disponible en: http://www.lgblog.cl/tecnologia/que-es-el-cache/ [Accedido el 31 Nov. 2017].

[51]. Lightnvm.io. (2018). LightNVM. [online] Disponible en: http://lightnvm.io/# [Accedido el 31 Jul. 2018].

[52]. Linuxcontainers.org. (n.d.). Linux Containers - LXD - Introduction. [online] Disponible en: https://linuxcontainers.org/lxd/introduction/ [Accedido el 30 Mar. 2018].

[53]. Luis, Á. (2015). SSD o eMMC en tablets, diferencias y similitudes - Alsitecno.com. [online] Alsitecno.com. Disponible en: http://alsitecno.com/2015/05/15/ssd-o-emmc-en-tablets-diferencias-y-similitudes/ [Accedido el 10 Dec. 2017].

[54]. Medium. (2018). How to train YOLOv3 to detect custom objects – Manivannan Murugavel – Medium. [online] Disponible en: https://medium.com/@manivannan_data/how-to-train-yolov3-to-detect-custom-objects-ccbcafeb13d2 [Accedido el 14 Aug. 2018].

[55]. Medium. (2018). Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3. [online] Disponible en: https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088 [Accedido el 26 Aug. 2018].

[56]. Medium. (2018). mAP (mean Average Precision) for Object Detection – Jonathan Hui – Medium. [online] Disponible en: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173 [Accedido el 27 Aug. 2018].

[57]. Medium. (2018). Top 15 Deep Learning applications that will rule the world in 2018 and beyond. [online] Disponible en: https://medium.com/@vratulmittal/top-15-deep-learning-applications-that-will-rule-the-world-in-2018-and-beyond-7c6130c43b01 [Accedido el 2 Sep. 2018].

[58]. Meneame.net. (2015). Canonical lanza LXD, el hipervisor más rápido del mundo [ENG]. [online] Disponible en: https://www.meneame.net/m/SysDevs/canonical-lanza-lxd-hipervisor-mas-rapido-mundo-eng [Accedido el 30 Mar. 2018].

[59]. Mipi.org. (2017). MIPI C-PHY. [online] Disponible en: https://mipi.org/specifications/c-phy [Accedido el 4 Dec. 2017].

[60]. Mipi.org. (2017). MIPI D-PHY. [online] Available at: https://mipi.org/specifications/d-phy [Accessed 4 Dec. 2017].

[61]. Mipi.org. (2018). MIPI Display Serial Interface (MIPI DSI). [online] Disponible en: https://mipi.org/specifications/dsi [Accedido el 10 Feb. 2018].

[62]. needed, w. (2012). why are separate icache and dcache needed. [online] Stack Overflow. Disponible en: https://stackoverflow.com/questions/8707041/why-are-separate-icache-and-dcache-needed [Accedido el 14 Nov. 2017].

[63]. NVIDIA Developer. (2018). CUDA Toolkit. [online] Disponible en: https://developer.nvidia.com/cuda-toolkit [Accedido el 31 Jul. 2018].

[64]. NVIDIA Developer. (2018). Linux For Tegra. [online] Disponible en: https://developer.nvidia.com/embedded/linux-tegra [Accedido el 29 Jul. 2018].

[65]. NVIDIA Developer. (2018). NVIDIA cuDNN. [online] Disponible en: https://developer.nvidia.com/cudnn [Accedido el 3 Aug. 2018].

[66]. NVIDIA Developer. (2018). NVIDIA TensorRT. [online] Disponible en: https://developer.nvidia.com/tensorrt [Accedido el 1 Aug. 2018].

[67]. NVIDIA Developer. (2018). VisionWorks. [online] Disponible en: https://developer.nvidia.com/embedded/visionworks [Accedido el 2 Aug. 2018].

[68]. NVIDIA Developer. (2018). Vulkan Support on L4T. [online] Disponible en: https://developer.nvidia.com/embedded/vulkan [Accedido el 1 Aug. 2018].

[69]. NVIDIA Jetson TX1/TX2 Developer Kit Carrier Board. (2017)..

[70]. NVIDIA Jetson TX2 System-on-Module Pascal GPU + ARMv8 + 8GB LPDDR4 + 32GB eMMC + WLAN/BT. (2017)..

[71]. NVIDIA's Next Generation CUDA Compute Architecture: Fermi. (2009). .

[72]. NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110/210. (2014). .

[73]. Nvidia.es. (2018). GPU aceleradora NVIDIA Tesla K80 para alta computación |NVIDIA. [online] Disponible en: https://www.nvidia.es/object/tesla-k80-dual-gpu-accelerator-oct-14-2014-es.html [Accedido el 31 Aug. 2018].

[74]. Nvidia.es. (2018). Las GPU ayudan a crear la mayor red neuronal artificial del mundo|NVIDIA. [online] Disponible en: https://www.nvidia.es/object/tesla-gpus-help-artificial-neural-network-es.html [Accedido el 31 Aug. 2018].

[75]. NVIDIA. (2018). Kits de desarrollo y Módulos Jetson para sistemas embebidos. [online] Disponible en: https://www.nvidia.es/autonomous-machines/embedded-systems-dev-kits-modules/ [Accedido el 15 Apr. 2018].

[76]. Nvidia.es. (2018). Procesamiento paralelo CUDA | Qué es CUDA |NVIDIA. [online] Disponible en: https://www.nvidia.es/object/cuda-parallel-computing-es.html [Accedido el 10 Mar. 2018].

[77]. NVIDIA TESLA V100 GPU ARCHITECTURE. (2017). .

[78]. Opencv.org. (n.d.). OpenCV library. [online] Disponible en: https://opencv.org/ [Accedido el 22 Aug. 2018].

[79]. Quora.com. (2015). What are the definitions of GPU chips, TPC & GPC? - Quora. [online] Disponible en: https://www.quora.com/What-are-the-definitions-of-GPU-chips-TPC-GPC [Accedido el 29 Nov. 2017].

[80]. PowerData, G. (2018). Big Data: ¿En qué consiste? Su importancia, desafíos y gobernabilidad. [online] Powerdata.es. Disponible en: https://www.powerdata.es/big-data [Accedido el 31 Aug. 2018].

[81]. Redes neuronales artificiales fundamentos y aplicaciones. (1993). .

[82]. RedesZone. (2016). ZFS, las características de este sistema de archivos avanzado. [online] Disponible en: https://www.redeszone.net/2016/10/01/zfs-las-caracteristicas-este-sistema-archivos-avanzado/ [Accedido el 2 Sep. 2018].

[83]. Redmon, J. (2018). YOLO: Real-Time Object Detection. [online] Pjreddie.com. Disponible en: https://pjreddie.com/darknet/yolo/ [Accedido el 15 Aug. 2018].

[84]. Redmon, J. (2018). YOLO: Real-Time Object Detection. [online] Pjreddie.com. Disponible en: https://pjreddie.com/darknet/yolov1/ [Accedido el 28 Aug. 2018].

[85]. RedUSERS. (n.d.). Cómo aprovechar nuestro Wi-Fi al máximo - RedUSERS. [online] Disponible en: http://www.redusers.com/noticias/como-aprovechar-nuestro-wi-fi-al-maximo/ [Accedido el 8 Dec. 2017].

[86]. Routerguide.net. (n.d.). DTIM Interval (Period) Best Setting | Router Guide. [online] Disponible en: https://routerguide.net/dtim-interval-period-best-setting/ [Accedido el 8 Dec. 2007].

[87]. Santos, L. (2018). Yolo · Artificial Inteligence. [online] Leonardoaraujosantos.gitbooks.io. Disponible en: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/single-shot-detectors/yolo.html [Accedido el 27 Aug. 2018].

[88]. Satran, M. (n.d.). Shader Model 4. [online] Docs.microsoft.com. Disponible en: https://docs.microsoft.com/en-us/windows/desktop/direct3dhlsl/dx-graphics-hlsl-sm4 [Accedido el 2 Sep. 2018].

[89]. Scantips.com. (2010). More details about Luminance in Histograms. [online] Disponible en: https://www.scantips.com/lumin.html [Accedido el 31 Aug. 2018].

[90]. Sgcg.es. (2016). Qué es el DRM - SGCG. [online] Disponible en: http://sgcg.es/articulos/2016/08/20/que-es-el-drm/ [Accedido el 10 Dec. 2017].

[91]. Sistop2013.blogspot.com. (2013). Memoria cache L1, L2 y L3. [online] Disponible en: http://sistop2013.blogspot.com/2013/04/memoria-cache-l1-l2-y-l3.html [Accedido el 12 Nov. 2017].

[92]. Thermal Design Guide. (2017)..

[93]. Timebutt.io. (2017). Understanding YOLOv2 training output. [online] Disponible en: https://timebutt.github.io/static/understanding-yolov2-training-output/ [Accedido el 16 Aug. 2018].

[94]. Towards Data Science. (2018). Training Object Detection (YOLOv2) from scratch using Cyclic Learning Rates. [online] Disponible en: https://towardsdatascience.com/training-object-detection-yolov2-from-scratch-using-cyclic-learning-rates-b3364f7e4755 [Accedido el 28 Aug. 2018].

[95]. Towards Data Science. (2018). What's new in YOLO v3? – Towards Data Science. [online] Disponible en: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b [Accedido el 17 Aug. 2018].

[96]. YOLOv3: An Incremental Improvement. (n.d.)..

[97]. You Only Look Once: Unified, Real-Time Object Detection. (2016)..

[98]. YUV 422, Y. (2011). YUV 422, YUV 420, YUV 444. [online] Stack Overflow. Disponible en: https://stackoverflow.com/questions/8561185/yuv-422-yuv-420-yuv-444 [Accedido el 15 Nov. 2017].

[99]. Cs.famaf.unc.edu.ar. (2018). Arquitectura de GPUs. [online] Disponible en: https://cs.famaf.unc.edu.ar/~nicolasw/Docencia/CP/8-gpuarch.html#slide41 [Accedido el 4 Sep. 2018].

[100]. Computerlanguage.com. (2018). SCSA - CDE Definition. [online] Disponible en: https://www.computerlanguage.com/results.php?definition=SCSA [Accedido el 1 Sep. 2018].

8.3. Otras referencias

8.3.1 Videos YouTube

[101]. Build OpenCV on NVIDIA Jetson TX2. (2017). [video].

[102]. Build OpenCV with CUDA on NVIDIA Jetson TX2. (2018). [video].

[103]. JetPack 3.0 - NVIDIA Jetson TX2. (2017). [video].

[104]. Package OpenCV Part 2 and YOLO!. (2018). [video].

[105]. Testing CUDA Programs on NVIDIA JETSON TX2. (2018). [video].

[106]. Ventajas y Desventajas De Instalar "UBUNTU". (2011). [video].

[107]. yolov3 custom object detection in linux mint or ubantu. (2018). [video].

8.3.2 Videos NVidia

[108]. NVIDIA Developer. (2018). Tutorials. [online] Disponible en: https://developer.nvidia.com/embedded/learn/tutorials [Accedido el 24 Feb. 2018].

8.3.3 Descarga de programas

[109]. ***Jetson-Inference:*** GitHub. (2018). dusty-nv/jetson-inference. [online] Disponible en: https://github.com/dusty-nv/jetson-inference [Accedido el 22 Jul. 2018].

[110]. ***Jetson-Reinforcement:*** GitHub. (2018). dusty-nv/jetson-reinforcement. [online] Disponible en: https://github.com/dusty-nv/jetson-reinforcement [Accedido el 29 Jul. 2018].

[111]. ***OpenCV 3.4.1:*** GitHub. (2018). jetsonhacks/buildOpenCVTX2. [online] Disponible en: https://github.com/jetsonhacks/buildOpenCVTX2 [Accedido el 17 Aug. 2018].

[112]. ***OpenCV 3.4.0:*** Jkjung-avt.github.io. (2018). How to Install OpenCV (3.4.0) on Jetson TX2. [online] Disponible en: https://jkjung-avt.github.io/opencv3-on-tx2/ [Accedido el 30 Jul. 2018].

[113]. ***YOLOv3:*** Jkjung-avt.github.io. (2018). YOLOv3 on Jetson TX2. [online] Disponible en: https://jkjung-avt.github.io/yolov3/ [Accedido el 8 Aug. 2018]

Capítulo 9

Lista de definiciones y acrónimos

[1]. ALU (UNIDAD ARITMÉTICA LÓGICA)

Es un circuito digital que como indica su nombre realiza operaciones aritméticas y lógicas. Es controlada mediante una CU, quien le indica que operación ha de realizar, siendo la entrada el conjunto de datos a operar. La CU extrae instrucciones de la memoria, las decodifica y las ejecuta, llamando a la ALU cuando sea necesario.

[2]. API (APPLICATION PROGRAMMING INTERFACE)

Es un conjunto de subrutinas y funciones que proporcionan bibliotecas para la programación de aplicaciones. Permite la comunicación entre distintos componentes software, sistemas operativos y protocolos de comunicación. Estas bibliotecas están formadas por un conjunto de reglas, lo que permite hacer uso de funciones ya existentes en otro software, ya que estos códigos están normalizados.

[3]. ARM

Es una familia de arquitecturas de instrucciones reducidas, tiene varias versiones en las que desde la versión 3 hasta la 7 los registros son fijos de 32 bits. Con la versión 8 aparece Thumb que admite registros variables de 32 y 64 bits. Está basado en RISC [48] por lo que requiere un menor número de transistores. Esto hace que se reduzca el calor obteniendo un mayor ahorro de energía. El conjunto de instrucción es ortogonal y cargan o almacenan diferentes arquitecturas.

- Arm 32 bits (Arquitectura Aarch 32): ejecuta las instrucciones A32 y T32. Este último formato de T32 no tiene equivalente de 64 bits, por lo que sólo puede acceder a los registros de 32 bits.
- ARM 64 bits (Arquitectura Aarch 64): ejecuta las instrucciones A64.

[4]. AXI (INTERFAZ EXTENSIBLE AVANZADO)

Interfaz entre host y device, con diferentes modos de conexión. Según el modo de interconexión se puede conectar uno o más maestros y se obtiene distintos niveles de rendimiento. Su uso está más orientado a sistemas con altas frecuencias de reloj y permite la transferencia de datos no alineados. Su estructura presenta fases separadas de dirección, control y datos. Para un mayor rendimiento del bus, el host puede realizar múltiples direccionamientos. Además, si se trata de un envió rápido sólo es necesario el inicio de la dirección.

[5]. ISM (INDUSTRIAL, SCIENTIFIC AND MEDICAL)

Son bandas de uso gratuito, sin necesidad de licencia, utilizadas por áreas industriales y científicas. Esta banda de radiofrecuencia electromagnética es sensible a errores, por lo que se utilizan mecanismo de protección contra interferencias.

[6]. CANONICAL

Es una empresa que da soporte software gratuito, siendo uno de los más importante el SO Ubuntu. Esta empresa comenzó siendo una organización virtual, en la que sus trabajadores programaban desde su casa, actualmente tiene a empleados trabajando en más de 30 países. (Es.wikipedia.org, 2018) [16]

[7]. CAR

Este término hace referencia al conjunto de palabras “Clock and Reset”.

[8]. CLÚSTER

Es una unidad compuesta por un conjunto de sectores colocados de manera contigua. Se utiliza para el almacenamiento en los discos. Dependiendo el tamaño de los archivos se almacenan en uno o más unidades. Hay varios tamaños de clúster siendo el más pequeño de 8 bytes, y el más grande de 512 bytes. Por ello un clúster puede contener a otro clúster de menor tamaño.

[9]. CMOS – SENSOR

El funcionamiento de este sensor es detectar la luz, mediante numerosos fotodiodos y un conversor digital. Se utiliza un fotodiodo por cada píxel, que dependiendo de la intensidad de luz detectada producen una corriente eléctrica. Su salida está conectada a un amplificador de señal eléctrica, produciendo un cuello de botella debido a la reducida velocidad de esta parte electrónica. Para mejorar esto se ha reducido el tamaño de los componentes electrónicos y se ha incorporado micro lentes que concentran la luz de cada celda en su fotodiodo. Este contiene un filtro para captar solo la luz roja, otra para la verde y otra para la azul. Obteniendo de este modo una mejor sensibilidad a la luz y una estructura más simple.

[10]. C-PHY

Se trata de una capa física que mejora el rendimiento en canales de ancho de banda limitado. Las transiciones entre los distintos modos son de baja latencia y cuenta con un reloj que reasigna las líneas dentro de un enlace. (Mipi.org, 2017) [59]

[11]. CRITOGRÁFIA

Consiste en técnicas de cifrados cuyo objetivo es proteger la información de los sistemas. Para ello se utilizan herramientas que comprueban si las condiciones son seguras o no.

[12]. CSMA/CD (CARRIER SENSE MULTIPLE ACCESS WITH COLLISION DETECTION)

Es un algoritmo de código abierto que se utiliza para mejorar las prestaciones de Ethernet. Este algoritmo lo que hace es que los dispositivos de red escuchen el medio antes de transmitir. De este modo se conoce previamente si el canal se encuentra libre para realizar la transmisión.

[13]. CUDA (Arquitectura Unificada de Dispositivos de Cómputo)

Es una arquitectura que realiza cálculo paralelo aprovechando así la gran potencia de la GPU para proporcionar un mayor rendimiento al sistema. Incluye un compilador y un conjunto de herramientas, desarrollada por NVIDIA. Además, cuenta con múltiples núcleos y es compatible con muchos lenguajes de programación.

[14]. D-Cache (Datos de Rápido acceso)

Conjunto de datos que son almacenados en una memoria de manera temporal. El acceso a estos datos se hace de manera instantánea y directa (LG, 2016).

[15]. DEBIAN

Es un SO libre formado por un conjunto de programas y funciones básicas. Es compatible con los núcleos Linux y Kfreebsd, y mantiene el software de manera sincronizado, estable y actualizado. Al contar con pocas funciones sus herramientas gráficas de configuración reducidas. Tiene varias versiones simultaneas que son estables, inestables, en prueba (testing) y experimentales. Este software se puede descargar completo e instalarlo posteriormente sin la necesidad de conectarse a internet. (Debian.org,2018) [9]

[16]. Direct3D 10

Es una API utilizada para la programación de gráficos 3D y es la principal competidora de OpenGL. Su principal aplicación es en los videojuegos, y su utilización facilita la creación de elementos gráficos y transformaciones geométricas. Una de sus principales características es su comunicación directa con los drivers, por lo que las representaciones de los gráficos en la pantalla son muy buenos. (Es.wikipedia.org, 2018) [17]

[17]. DMA (Direct Memory Access)

Permite acceder a los datos de la memoria del sistema de forma directa, independientemente de la CPU. De este modo permite a los dispositivos de diferentes velocidades comunicarse con la memoria.

[18]. DP (Display Port)

Es un interfaz de conexión digital utilizado en la conexión de dispositivos para la transmisión de vídeos y audios. Soporta 1, 2 y 4 pares de datos en el enlace principal y un máximo de 24 bpp en el caso de una señal de vídeo. Mientras que si se trata de una señal de audio soporta hasta ocho canales de 192 KHz sin compresión. Su longitud máxima de transmisión de datos es de 15 metros y la versión 1.2 se caracteriza por el aumento de su ancho de banda a 17.28 Gbit/s, aumentando así la resolución, la tasa de actualización y la profundidad de color.

[19]. D-PHY

Capa física que proporciona una alta inmunidad al ruido y tolerancia a la fluctuación de fase. Las transiciones entre lo modos son de baja latencia (Mipi.org, 2017) [60]

[20]. DRM

Este término se refiere a las tecnologías de control de acceso, su función es restringir la conexión de medios o dispositivos digitales a equipos no autorizados. (Sgcg.es, 2016) [90]

[21]. DRAM (MEMORIA DINÁMICA DE ACCESO ALEATORIO)

Es un tipo de tecnología de memoria RAM que se basa en un transistor de efecto de campo y un condensador. El condensador pierde su carga de forma progresiva, por lo que está conectado con un circuito dinámico de refresco. Este circuito integra un transistor por lo que revisa cada cierto tiempo la carga y la repone en un ciclo de refresco.

[22]. DSI (INTERFAZ SERIAL INTERFACE)

Es una interfaz específica de MIPI y su bus de transferencia de datos tiene una línea de reloj de alta velocidad y uno o más carriles. Estos carriles son direccióneles y se transportan mediante dos hilos, desde el host al device. Excepto la primera línea de datos, la cual es bidireccional y puede invertir la dirección de transmisión (Mipi.org, 2017) [61]

[23]. DTIM (DELIVERY TRAFFIC INDICATION MESSAGE)

Es un mensaje que indica según el número recibido si puede entrar o no en modo de ahorro de energía. En el caso de que el número sea reducido, no se pueden poner en modo ahorro, por el contrario, si en número es elevado sí que podrán. Hay cierta información que les obliga a estar activos y no permite el modo de ahorro de energía, por lo que se le ha de comunicar su activación (RedUSERS, n. d.) [85] y (Routerguide.net, n. d.) [86]

[24]. EEPROM

Este dispositivo tiene forma de tarjeta, que se utiliza para almacenar grandes cantidades de datos en un espacio reducido. Se caracteriza por almacenar sin la necesidad de estar conectada a una fuente de alimentación, además permite la lectura y reescritura de múltiples posiciones de memoria en la misma operación.

[25]. EMMC (EMBEDDED MULTIMEDIACARD)

Este módulo integrado está formado por una memoria y un controlador de memoria flas. Se caracteriza por ser un sistema de memoria no volátil y simplificar el diseño de la interfaz de las aplicaciones (Kingston Technology Company, n.d.) [44]

[26]. GDDR3 (GRAPHICS DOUBLE DATA RATE 3)

Es una tecnología de memoria RAM utilizada en tarjetas gráficas, que es capaz de transferir por ciclo de reloj un total de 64 bits, soportan una máxima capacidad de 1 GB con un ancho de banda entre 166 y 800 MHz. (Es.wikipedia.org, 2018) [18]

[27]. GDDR5 (GRAPHICS DOUBLE DATA RATE 5)

Memoria muy utilizada en las tarjetas gráficas, es de acceso aleatorio y combina la optimización de información, el interfaz y la compensación de errores. A diferencia de GDDR3 esta presenta menor coste, mayor rendimiento y mayor ancho de banda con una interfaz de memoria mucho más pequeña. (Es.wikipedia.org, 2018) [19]

[28]. MEMORIA FIFO

El modo de funcionamiento de esta memoria se caracteriza por sacar los datos en el orden en el que han sido introducidos. Se utiliza para almacenar y controlar el flujo de transmisión de datos mediante punteros. Su transmisión puede ser síncrono o asíncrono y tiene cuatro estados que son casi lleno, lleno, vacío y casi vacío.

[29]. GPC (NÚCLEO DE PROCESAMIENTO GRÁFICO)

Son núcleos contenidos en la GPU que se utilizan para la rasterización de imágenes y está formado por unidades de texturas, SFU, PolyMorph y hasta por cuatro SM. (Quora.com, 2015) [79]

[30]. GPIO (GENARAL PURPOSE INPUT/OUTPUT)

Es un pin genérico que se puede controlar durante el tiempo de ejecución. Su configuración se realiza mediante MPIO, este contiene la información hardware necesaria para optimizar la conectividad con las matrices de almacenamiento de datos. La configuración puede ser de forma individual para cada pin, configurando como salida, entrada o interruptor. Esta configuración se realiza con controles de nivel y borde.

[31]. HBM2 (HIGH BANDWITH MEMORY)

Se trata de una interfaz RAM de alto rendimiento que obtiene un mayor ancho de banda al utilizar menos energía. Esto se consigue apilando matrices DRAM, cuyo controlador de memoria esta conectadas a estas mediante uniones de silicios. Esta versión cuenta con ocho matrices DRAM por columna y un ancho de banda de 1024 bits en total. (En.wikipedia.org, 2018) [20]

[32]. HD (HIGH DEFINITION)

Hace referencia a un sistema de imagen, sonido o video en el que se alcanza la resolución máxima de 1920x1080 píxeles, superando de este modo a la resolución estándar que es de 720x576 píxeles en el sistema PAL.

[33]. HIPERVISOR LXD

Este hipervisor ha sido desarrollado por Canonical y se encarga de la administración de los contenedores del sistema LINUX. Este sistema es seguro y de alto rendimiento al ser de baja latencia y muy rápido. (Meneame.net, 2015) [58] y (Linuxcontainers.org, n.d.) [52]

[34]. HYPER-Q

Esta arquitectura permite aumentar el número de procesos realizados por la CPU, por lo que se consigue de este modo el funcionamiento simultaneo de la CPU con la GPU. Esto reduce los tiempos inactivos de la GPU y aumenta de este modo el número de conexiones simultáneas. El uso de esta arquitectura es flexible ya que se utiliza para conexiones de transmisiones CUDA o en procesos de interfaz de paso de mensajes (MPI). (Hyper-Q Example, 2013) [36]

[35]. I-CACHE (MEMORIA DE CACHE DE INSTRUCCIONES)

Para evitar el riesgo estructural en segmentación, se utiliza una memoria para almacenar las instrucciones separando de este modo el almacenamiento de los datos e instrucciones. Con esto se aumenta el rendimiento y se reduce el consumo de energía (needed, 2012) [62]

[36]. KERAS API

Esta API es de redes neuronales de alto nivel que se utiliza para modelos complejos en los que las salidas son múltiples, los gráficos son acíclicos o las capas son compartidas. Su lenguaje es Python y es compatible con TensorFlow, CNTK y Theano. Facilita el desarrollo de prototipos o la reutilización de modelos entrenados y además es compatible con redes neuronales convolucionales y recurrentes. (Keras. Io, n. d.) [42] [43]

[37]. L1-L2 (NIVEL 1 – NIVEL 2)

La memoria del sistema esta dividía en un conjunto de bloques que están ordenados por niveles. El controlador y procesador están colocados en el nivel más rápido que es el L1. Este está dividido a su vez en dos partes, por un lado, están los datos que es D-Cahe y por otro está la I-Cache que almacena las instrucciones.

El siguiente nivel es el L2, a diferencia del anterior este no distingue entre datos e instrucciones, por lo que es más lento que L1 y mucho más grande. (Sistop2013.blogstop.com,2013) [91]

[38]. LIGHTNVM

Es un subsistema de SSD de canal abierto de Linux que permite el ajuste de la latencia de lectura y con el que las latencias son predecibles. Además, con la asignación por separado del SSD permite aislar las entradas y salidas del sistema (Lightnvm.io, 2018) [51]

[39]. MEMORIA CACHE

Es una memoria ultrarrápida que emplea el procesador para tener alcance directo a ciertos datos que posiblemente serán utilizados en las siguientes operaciones, sin tener que acudir de este modo a la memoria principal, reduciendo así el tiempo de espera. Funciona como una memoria auxiliar, la cual es mucho más rápida y de menor tamaño que la memoria principal. Los micros compatibles con PC poseen la caché interna en L1, mientras que los más modernos, la incluyen en L2, que es más grande y lenta, incluso en algunos existe L3. Cuando se acede por primera vez a un dato, se hace una copia en esta memoria, por lo que en las siguientes ejecuciones se accederá a esta, de este modo se consigue que sea de menor tiempo el acceso a los datos. Esta memoria se suele situar en la CPU y la RAM (Memoria de Acceso Aleatorio), para acelerar de este modo el intercambio de datos. (LG, 2016) [50]

[40]. MIPI (MOBILE INDUSTRY PROCESSOR INTERFACE)

Se trata de un interfaz serie dirigido por Alliance que se caracteriza por ser de alta velocidad entre el procesador host y el módulo de visualización.

[41]. MMIO (MEMORIA MAPEADA DE ENATRADA/SALIDA)

Es un método utilizado para direccionar la CPU con las E/S de los periféricos, se caracteriza por utilizar el mismo bus de direcciones para memoria y dispositivos de E/S. Esto se debe a que las instrucciones utilizadas por la CPU para el acceso a la memoria son las mismas que para acceder a los dispositivos periféricos. La diferencia entre ellos es que en el caso de los dispositivos de E/S el espacio es reservado por la CPU de modo temporal o permanente. Por lo que cada dispositivo de E/S habilita el bus de direcciones de la CPU y responde a cualquier acceso de esta, conectando el bus de datos con la localización en memoria física del dispositivo deseado.

[42]. MSELECT FIFO

Es el elemento donde la información de entrada se almacena, ya que atiende a la primera llamada, por lo que el resto tienen que esperar y colocarse al final de la cola de entrada. Atenderá a las llamadas según el orden de llegada.

[43]. MSS (TAMAÑO MÁXIMO DE SEGMENTO)

Se presentan en bytes y es el tamaño más grande de datos que un dispositivo puede recibir en un único trozo, sin fragmentar. Para que la comunicación sea óptima, la suma de bytes sin fragmentar y la cabecera ha de ser menor que el número de bytes de la MTU (Unidad Máxima de Transferencia).

[44]. NÚCLEOS SIP (SYSTEM IN A PACKAGE)

Son un conjunto de circuitos integrados que se encuentran en un mismo módulo que se encarga de realizar la mayoría de las funciones del sistema. A diferencia de los módulos multichip estos se pueden colocar en ambas posiciones y no sólo en horizontal, y une los diferentes circuitos mediante uniones de cables o soldaduras. La desventaja de usar estos tipos de módulos es que si uno de sus componentes no funciona el módulo completo es por tanto defectuoso.

[45]. OPENGL 2.1

Esta versión de OpenGL fue publicada en 2006 y es compatible con las versiones anteriores. Las mejoras que ofrece esta versión con respecto a las anteriores es que incluye un nuevo comando para acelerar la transmisión de imágenes y otro para trabajar con las texturas de las imágenes en RGB. (Es.wikipedia.org, 2018) [23]

[46]. PMC (PRODUCTION MONITORING CONTROL)

Este diseño de control de energía permite establecer algunas aplicaciones en modo de suspensión durante el tiempo de ejecución. Este sistema incluye una máquina de estado finito (FSM) simple para administrar el modo de estado de baja potencia al apagar el búfer de E/S y la activación de GCLK durante el modo de suspensión. Esto le permite desactivar partes del diseño, reduciendo así el consumo dinámico de energía. La habilitación de la aplicación se realiza de forma rápida con una activación de menos de 1 ms.

[47]. PLLS (PHASE LOCKED LOOP)

Son dispositivo cuyos sistemas de frecuencia y fase son realimentados, y existen dos modos de analizarlos, según se considere que los componentes son lineales o no. De este modo existe el análisis lineal en el que se consideran que, si son lineales todos los componentes, y el análisis no lineal en el que los dispositivos se consideran independientemente tal cual son. Estos dispositivos presentan cuatro zonas de funcionamiento que son las siguientes:

- Hold in: En este el PLL continúa enganchado, por lo que puede seguir suministrando frecuencias muy lentas.
- Pull out: En esta zona el PLL no se desengancha aun con un salto brusco de entrada.
- Lock in: En esta zona el PLL está desenganchado y puede engancharse en un periodo menor de la frecuencia de la señal de salida.

- Pull in: A diferencia de la anterior el enganche del PLL se da en un periodo más largo que la de la señal de salida.

[48]. RISC (REDUCED INSTRUCTION SET COMPUTER)

Es una arquitectura utilizada en las CPU que cuenta con menos tipos de instrucciones y de tamaño fijo, por lo que su velocidad es mayor y ejecuta más instrucciones por segundo. El rendimiento de esta arquitectura es mayor debido a que elimina las instrucciones innecesarias, en las que únicamente las instrucciones de carga y almacenamiento tienen acceso a la memoria de datos, y además las rutas de acceso están optimizadas.

[49]. SCSA (SIGNAL COMPUTING SYSTEM ARCHITECTURE)

Arquitectura estándar para los componentes hardware y software en sistemas de transmisión de señales de voz y vídeo. (Computerlanguage.com, 2018) [100]

[50]. SHADERS

Son pequeños programas definidos por el usuario que son ejecutados por la GPU y permiten que el programador interaccione con el sistema mediante la modificación de parámetros. Estos programas están compuestos por un conjunto de bloques que realizan un determinado procesamiento y lo pasan al siguiente bloque hasta sacar la imagen por pantalla. Tienen su propio lenguaje de programación y las principales funciones que realizan son el posicionamiento de vértices en la pantalla, el cálculo del color, la rasterización y el texturizado. (for?, 2013) [29] (Learnopengl.com, 2014) [49]

[51]. SHADERS UNIFICADOS (UNIFIED SHADING ARCHITECTURE)

Hace referencia a dos conceptos: una es la arquitectura que presenta el shader y el otro al conjunto e instrucciones utilizadas. Si se trata de la arquitectura unificada esta se refiere a la realización de varias operaciones con el mismo hardware, evitando así que la sobrecarga de trabajo. En el caso del modelo unificado hace referencia a que las instrucciones utilizadas son las mismas para todos los bloques, teniendo además cada uno las suyas propias. (Es.wikipedia.org, n,d) [24]

[52]. SHADERS 4.0

Es el concepto utilizado por Direct3D para hacer referencia al modelo shader unificado, el cual se refiere a que todos los bloques del shader utilizan el mismo tipo de instrucciones para realizar las operaciones. Incorpora un total de cinco etapas de sombreado que son vértice, geometría, control y evaluación de teselación y fragmento. (Satran, n. d.) [88]

[53]. SM (MULTIPROCESADORES STREAMING)

Es un conjunto de hilos que están presentes en la GPU y en el que cada SM tiene su propia memoria interna lo que le permite una comunicación mucho más rápida. Se caracteriza por tener un elevado ancho de banda y baja latencia, por ello los datos son copiados en cada SM para acelerar así el procesamiento de datos.

[54]. SMX

La mejora que incorpora son los cuatro bloques de distribución de trabajo y las ocho unidades de distribución de instrucciones. Cada SMX está compuesto por 192 núcleos e incorpora 32 SFU y 32 unidades LD / ST. La distribución cuádruple permite que se puedan enviar dos instrucciones independientes por warp.

[55]. SoC (System on-Chip)

Hace referencia a un chip en el que se integran todos o la mayoría de los componentes o módulos de un sistema electrónico. Son muy utilizados en aplicaciones basados en sistemas embebidos y a diferencia de los microcontroladores el tamaño de las memorias es más grande. Además, son muy potentes capaces de procesar arquitecturas complejas.

[56]. SPs (Streams Processor)

Estos surgieron de la unificación de los shaders y hace referencia a un conjunto de procesadores que procesan la información de modo paralelo. Este conjunto de procesadores está unido mediante hilos por lo que cada uno puede trabajar a diferente frecuencia de reloj y obtiene mejores resultados al aprovechar toda la información procesada. (Advances in Cryptology-ASIACRYPT 2007, 2007) [3]

[57]. SPI (Serial Peripheral Interface)

Es un protocolo de comunicación síncrono que utiliza un reloj para regular el flujo de bits transmitido. Esto permite reducir el tamaño y número de pines necesarios para llevar a cabo la transferencia de datos. El modo de transferencia es full-duplex lo que permite mayor velocidad de transmisión y una comunicación simultánea utilizando diferentes canales.

[58]. TEGRA

Es un sistema chip desarrollado por NVIDIA, que da soporte a dispositivos portátiles y móviles para la conectividad a Internet. Estos contienen en un único paquete los siguientes componentes la CPU, GPU y el controlador de memoria, todo ello destaca por su bajo consumo de energía y alto rendimiento en reproducciones de audio y video.

[59]. Teselado

El teselado es un patrón de figuras que recubren por completo una superficie plana mediante transformaciones isométricas que se realizan sobre la figura inicial. Es decir, consiste en realizar copias idénticas con las cuales se componen figuras para recubrir la superficie.

[60]. TLB (Traslation Lookaside Buffer)

Es un buffer o memoria que es controlada por la unidad de gestión de memoria (MMU), quien se encarga de relacionar las direcciones físicas y lógicas. Su número de entradas es fijo y se utiliza para realizar una traducción rápida de direcciones en las que hay dos tipos. Por un lado, está el direccionamiento virtual en el que las peticiones son enviadas desde la CPU a la memoria directamente, y el otro tipo es el direccionamiento físico. En este segundo la CPU accede en cada operación al TLB, y la dirección física obtenida es enviada a la memoria caché.

[61]. TMDS (TRANSITION MINIMIZED DIFFERENTIAL SIGNALING)

Esta tecnología se caracteriza por su baja interferencia electromagnética entre sus cables de cobre gracias a su algoritmo de codificación. Esta codificación se realiza en dos etapas en la que convierte una entrada de 8 bits en una de 10 bits. Además, los datos son transmitidos en serie a alta velocidad y permite la recuperación del reloj en el receptor.

[62]. TMP (TOTAL MODULE POWER)

Hace referencia a la disipación de potencia promedio de la placa mientras el sistema ejecuta la carga de trabajo objetivo en las peores condiciones de estado estacionario.

[63]. TPU (TENSOR PROCESSING UNIT)

Se trata de un circuito integrado de tamaño reducido que ha sido desarrollado por Google para el desarrollo las aplicaciones basadas en Deep Learning. Estas unidades se caracterizan por dar soporte a mayor volumen de cálculos de precisión reducida y a nivel de procesamiento de imágenes es capaz de procesar más de 100 millones de fotos al día. (Es.wikipedia.org, 2018) [26]

[64]. UTP (UNSHIELDED TWISTED PAIR)

Son cables trenzados no apantallados por lo que se producen más errores, no se pueden utilizar para largas distancias ya que tienen un límite máximo y su impedancia es de 100 Ω.

[65]. VULKAN

Es una API multiplataforma, ya que es compatible con una amplia cantidad de SO, la cual es utilizada en aplicaciones con gráficos 3D (NVIDIA Developer, 2018) [68]. Su principal característica es que aprovecha todos los núcleos presentes en el procesador del PC, incrementando de este modo el rendimiento gráfico de la aplicación. Otras de sus características son:

- Reduce la carga de trabajo de la CPU, gracias al procesamiento por lotes, lo que le permite reducir la sobrecarga del controlador.
- Tiene mayor velocidad, ya que en el momento de trabajar con shaders estos ya están traducidos a un formato binario intermedio denominado SPIR-V.
- No es necesario de una API adicional de cálculo, ya que los núcleos y shaders se tratan de manera unificada.

[66]. WAP-AP (PROTOCOL APPLICATION WIRELESS)

Protocolo internacional utilizado en aplicaciones con comunicación inalámbrica.

[67]. XBAR

Rastrea y controla las características del host.

[68]. YCBCR

Este formato es una forma de codificar información RGB, por lo que no representa un color absoluto y se utiliza en sistemas de vídeo y fotografía digital. La codificación consiste en mostrar una señal como resultado de la combinación RGB. Cuyas siglas significan:

- Y: es la componente que codifica la luminosidad de la imagen.

- Cb: componente de crominancia diferencia de azul.
- Cr: componente de crominancia diferencia de rojo.

[69]. YOLO (YOU ONLY LOOK ONCE)

Es un algoritmo basado en el aprendizaje profundo de redes neuronales que detecta objetos en tiempo real. Para su entrenamiento se utilizan un conjunto de imágenes en la que los objetos aparecen en distintas posiciones y a diferentes escalas, en la que se ha de localizar y clasificar con etiquetas estos objetos. Una vez realizado el entrenamiento de la red neuronal el sistema será capaz de detectar aquellos objetos aprendidos (Redmon, 2018) [83]

[70]. YUV 222 - YUV 444

Es un sistema de interpretación del color en el que el color de la imagen o vídeo es codificado en función de la percepción humana. Por lo que el ancho de banda de diferencia de color es reducido, haciendo que los errores de transmisión sean más desapercibidos que si se utiliza el formato RGB.

Los números que aparecen colocados a la derecha de YUV indica la frecuencia de muestreo de cada componente. Por lo que YUV 444 indica que la profundidad de las tres componentes es de dos bytes, mientras que YUV 222 es de un byte. (YUV 422, 2011) [98]

[71]. ZFS (ZETTABYTE FILE SYSTEM)

Es un tipo de archivo que fue desarrollado por el SO Solaris en 2004, se caracteriza por su gran capacidad y por ser de código abierto. Cuenta además con medidas de protección de datos, evitando así la pérdida o corrupción de estos. (RedesZone, 2016) [82]

Anexos

Tabla de contenido

Shaders

Hace veinte años era la CPU quien se encarga de realizar las funciones de transformación que se cometían a los píxeles hasta llegar a su visualización. Estas transformaciones realizadas no eran programables, por lo que el programador no podía modificar parámetros como el color o textura. Pero esto cambio con la creación de los shaders, que son programas que pueden ser ejecutados independientemente, con los cuales era posible que el programador interaccione con el proceso. Estos programas se realizan con un leguaje específico entre los que destacan:

- GLSL: Su diseño está basado en el lenguaje C y se utiliza en el entorno de OpenGL.
- CG: Este lenguaje pertenece a NVIDIA y depende del sistema hardware, por lo que se ha de crear perfiles específicos para cada tarjeta gráfica.

Los shaders eran procesados por la GPU y dos de sus bloques principales son Vertex y Fragment shader, que realizan las siguientes funciones.

- Vertex shader: Se encarga de calcular la posición de los vértices en la pantalla y de preparar estos datos. Cada vértice contiene una determinada información por lo que Vertex shader se ejecutará tantas veces como vértices tenga, ya que cada vez que se ejecuta accede a un vértice diferente.
- Fragment shader: Calcula el color que debe aparecer por pantalla.

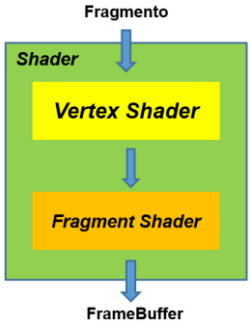


Figura 159: Estructura Shader

En el paso entre Vertex y Fragment la GPU calcula los píxeles que hay entre los vértices que Vertex ha determinado, para que posteriormente Fragment determine su color. Una vez realizada esta transformación la información se transfiere al Framebuffer que es donde se guarda la información que va a ser representada por pantalla. (Cs.famaf.unc.edu.ar, 2018) [99]

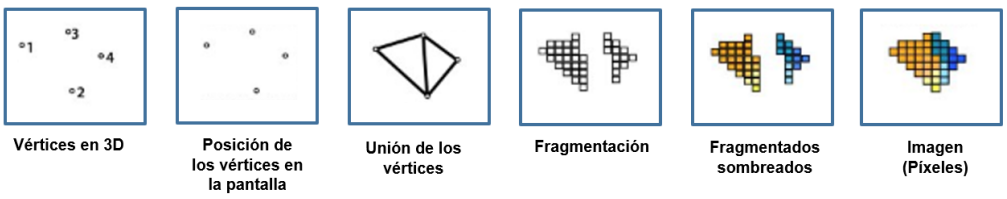


Figura 160: Procesos realizados Shader

El desarrollo de las GPUs dio lugar a los shaders unificados que hace referencia a dos conceptos distintos, uno es el modelo de instrucciones y el otro es la arquitectura.

El modelo shader unificado se refiere a que todos los bloques del shader utilizan el mismo tipo de instrucciones para realizar las operaciones. Se ha de tener en cuenta que además de estas instrucciones los bloque tendrán sus instrucciones específicas. Sin embargo, con el modelo tradicional cada bloque del shader tenía sus respectivas instrucciones. Este concepto en OpenGL se denomina *Modelo shader* mientras que en Direct3D es *Shader Model 4.0*.

La arquitectura shader hace referencia a la distribución que presenta la parte hardware que ejecuta los shaders. En la arquitectura original cada una de las operaciones se realizaba con una parte hardware específica, por lo que la parte que se encargaba de realizar una determina operación no intervenía en el resto de los procesos.

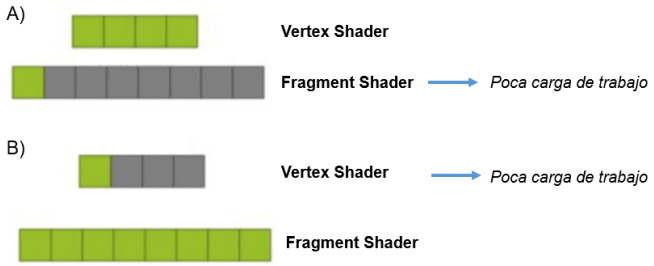


Figura 161: Shader

Con este modo de ejecución existía un desequilibrio en la carga de trabajo, por lo que se producían sobrecargas en determinadas partes hardware del sistema, mientras que otros componentes tenían poca carga y se desperdiciaba su funcionamiento.

La principal característica de la arquitectura unificada shader es que permite realizar varias operaciones o transformaciones con el mismo hardware. De este modo se elimina las particiones estáticas que presentaban los anteriores, se aumenta su funcionalidad y se obtienen mejores resultados.

Figura 162: Shader unificado

Arquitectura Tesla

El ejemplo de esta arquitectura es el procesador C1060 el cual está compuesto por 240 núcleos CUDA, la siguiente imagen muestra el aspecto que presenta esta arquitectura.

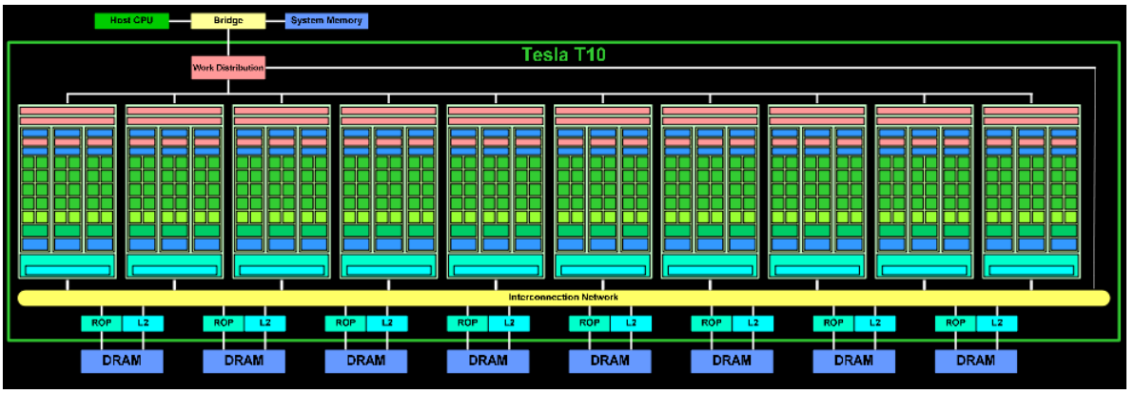


Figura 163: Arquitectura Tesla

Se observa que cada SM está compuesto por 8 núcleos CUDA, y por tanto hay 30 SM en total, distribuidos en 10 GPC.

Arquitectura Fermi

El ejemplo de la siguiente arquitectura es el procesador M2090 el cual está compuesto por 512 núcleos CUDA, la siguiente imagen muestra el aspecto que presenta esta arquitectura. (NVIDIA's Next Generation CUDA Compute Architecture: Fermi, 2009) [71]

Figura 164: Arquitectura Fermi

Esta arquitectura está compuesta por un total de 16 SM con 32 núcleos en cada uno.

Arquitectura Kepler

El procesador es el K80 el cual está compuesto por dos módulos GK210, que presenta la siguiente estructura. (NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110/210, 2014) [72]

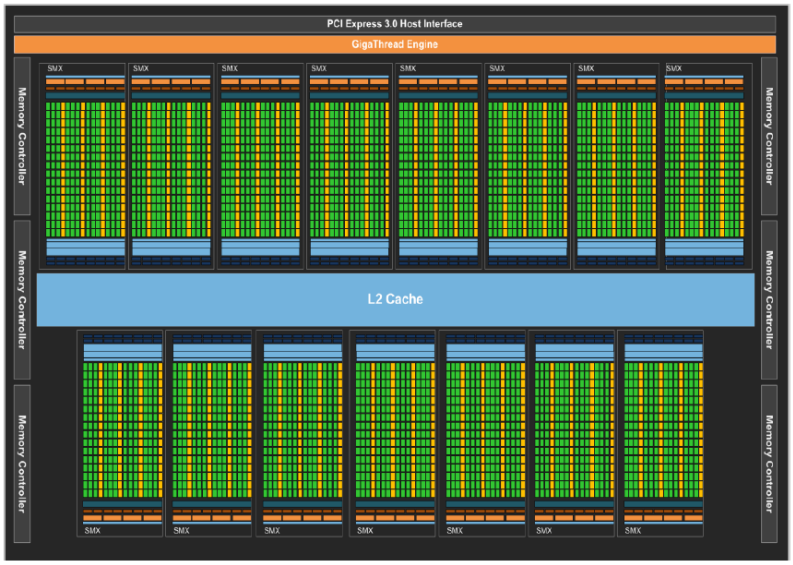


Figura 165: Arquitectura Kepler

Formado por 15 SMX y cada uno de estos por 192 núcleos. Cada SMX presenta cuatro bloques de distribución de trabajo y ocho unidades de distribución de instrucciones, permitiendo de este modo ejecutar cuatro warps al mismo tiempo. En la siguiente imagen se muestra una ampliación del SMX.

Figura 166: Distribución SMX

Además, posee 32 SFU (unidades de función especial) y 32 LD/ST (unidades de carga y almacenamiento.

Arquitectura Maxwell

El ejemplo de esta arquitectura es el procesador M60 el cual está compuesto por 2 módulos GM204, los cuales incorporan un total de 2048 núcleos. Su arquitectura a diferencia de la versión anterior, cada SM se divide en cuatro cuadrantes y cada uno contiene 32 SPs o núcleos.

Figura 167: Arquitectura Maxwell

En la siguiente imagen se muestra una ampliación del SMM, el cual posee un total de 32 SFU y 32 LD/ST, divididos en los cuatro bloques.

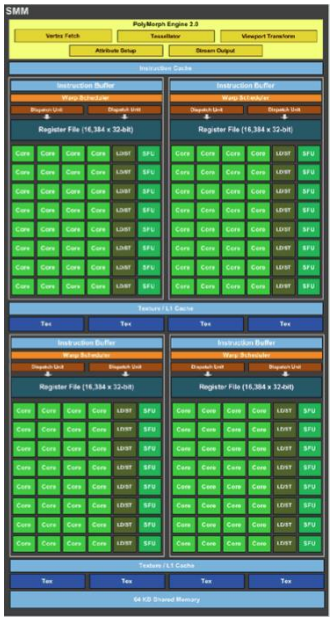


Figura 168: Distribuación SMM

Arquitectura Pascal

La arquitectura tiene 6 GPC las cuales controlan a 10 SM cada una, y está compuesta por un total de 3584 núcleos CUDA. Con la incorporación de HBM2 los bloques de memoria se disponen en dos bloques verticales unidos mediante matrices a los SM. Los bloques que se encuentran más externos representan a HBM2, y las dos columnas verticales que se encuentran en los laterales a las memorias.

Figura 169: Arquitectura Pascal

La siguiente imagen muestra la distribución que presenta cada SM, los cuales incorporan 32 núcleos cada uno, colocados en dos columnas verticales. En este caso incorporan 16 SFU y 16 LD/ST por SM.

Figura 170: Distribución SM

Arquitectura Volta

Es la última arquitectura presentada por NVIDIA la cual incorpora 5120 núcleos y es la idónea para realizar aplicaciones Deep Learning de grandes redes. (NVIDIA TESLA V100 GPU ARCHITECTURE, 2017) [77]

Figura 171: Arquitectura Volta

Las dos principales características de esta arquitectura es la unión de la memoria compartida con los recursos de L1, y la incorporación de los Tensor Core. Cada uno realiza 64 operaciones FMA de coma flotante, por lo tanto, ocho núcleos Tensor en un SM realizan 512 operaciones. Esta arquitectura contiene un total de 640 núcleos tensores, los cuales están distribuidos cada ocho por SM.

Figura 172: Distribución SM

INA3221

Las datasheet que se muestran a continuación corresponden al sistema de control de tensión INA3221 de TEXAS INSTRUMENTS. (INA3221 Triple-Channel, High-Side Measurement, Shunt and Bus Voltage Monitor with I 2C- and SMBUS-Compatible Interface, 2016) [38]

TEXAS INSTRUMENTS
INA3221
SBOS576B – MAY 2012 – REVISED MARCH 2016

INA3221 Triple-Channel, High-Side Measurement, Shunt and Bus Voltage Monitor with I²C- and SMBUS-Compatible Interface

1 Features

- Senses Bus Voltages From 0 V to 26 V
- Reports Shunt and Bus Voltage
- High Accuracy:
  - Offset Voltage: ±80 µV (max)
  - Gain Error: 0.25% (max)
- Configurable Averaging Options
- Four Programmable Addresses
- Programmable Alert and Warning Outputs
- Power-Supply Operation: 2.7 V to 5.5 V

2 Applications

- Computers
- Power Management
- Telecom Equipment
- Battery Chargers
- Power Supplies
- Test Equipment

3 Description

The INA3221 is a three-channel, high-side current and bus voltage monitor with an I²C- and SMBUS-compatible interface. The INA3221 monitors both shunt voltage drops and bus supply voltages, in addition to having programmable conversion times and averaging modes for these signals. The INA3221 offers both critical and warning alerts to detect multiple programmable out-of-range conditions for each channel.

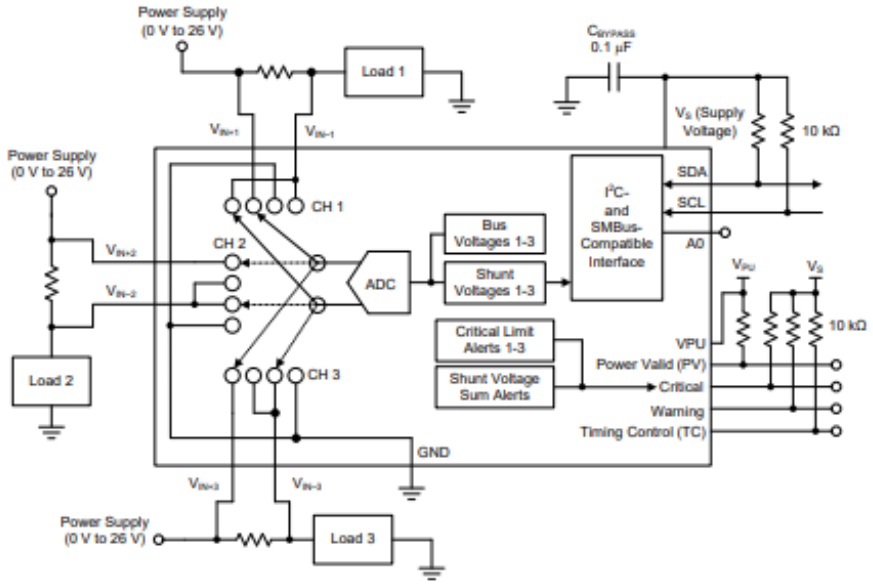
The INA3221 senses current on buses that can vary from 0 V to 26 V. The device is powered from a single 2.7-V to 5.5-V supply, and draws 350 µA (typ) of supply current. The INA3221 is specified over the operating temperature range of –40°C to +125°C. The I²C- and SMBUS-compatible interface features four programmable addresses.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
INA3221	VQFN (16)	4.00 mm x 4.00 mm

(1) For all available packages, see the package option addendum at the end of the datasheet.

Typical Application

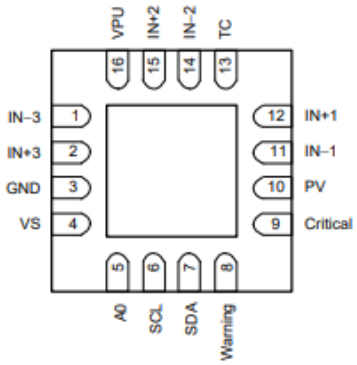


5 Device Comparison Table

DEVICE	DESCRIPTION
INA226	36-V, Bidirectional, Ultrahigh Accuracy, Low- or High-Side, I²C Out, Current and Power Monitor With Alert
INA219	26-V, Bidirectional, Zero-Drift, High-Side, I²C Out, Current and Power Monitor
INA209	26-V, Bidirectional, Low- or High-Side, I²C Out, Current and Power Monitor and High-Speed Comparator
INA210, INA211, INA212, INA213, INA214	26-V, Bidirectional, Zero-Drift, High-Accuracy, Low- or High-Side, Voltage Out, Current Shunt Monitor

6 Pin Configuration and Functions

RGV Package
16-Pin VQFN
Top View



Pin Functions

PIN		I/O	DESCRIPTION
NAME	NO.		
A0	5	Digital input	Address pin. Connect to GND, SCL, SDA, or V_S. Table 1 shows pin settings and corresponding addresses.
Critical	9	Digital output	Conversion-triggered critical alert; open-drain output.
GND	3	Analog	Ground
IN–1	11	Analog input	Connect to load side of the channel 1 shunt resistor. Bus voltage is the measurement from this pin to ground.
IN+1	12	Analog input	Connect to supply side of the channel 1 shunt resistor.
IN–2	14	Analog input	Connect to load side of the channel 2 shunt resistor. Bus voltage is the measurement from this pin to ground.
IN+2	15	Analog input	Connect to supply side of the channel 2 shunt resistor.
IN–3	1	Analog input	Connect to load side of the channel 3 shunt resistor. Bus voltage is the measurement from this pin to ground.
IN+3	2	Analog input	Connect to supply side of the channel 3 shunt resistor.
PV	10	Digital output	Power valid alert; open-drain output.
SCL	6	Digital input	Serial bus clock line; open-drain input.
SDA	7	Digital I/O	Serial bus data line; open-drain input/output.
TC	13	Digital output	Timing control alert; open-drain output.
VPU	16	Analog input	Pull-up supply voltage used to bias power valid output circuitry.
VS	4	Analog	Power supply, 2.7 V to 5.5 V.
Warning	8	Digital output	Averaged measurement warning alert; open-drain output.

7 Specifications

7.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)

			MIN	MAX	UNIT
Voltage	Supply, V_S			6	V
Analog inputs	IN+, IN–	Differential (V_{IN+}) – ($V_{IN–}$)(2)	–26	26	V
		Common-mode (V_{IN+}) + ($V_{IN–}$) / 2	–0.3	26	
	VPU			26	
Digital outputs	Critical, warning, power valid			6	V
	Timing control			26	
Serial bus	Data line, SDA		(GND – 0.3)	6	V
	Clock line, SCL		(GND – 0.3)	(V_S + 0.3)	
Current	Input, into any pin			5	mA
	Open-drain, digital output			10	
Temperature	Operating, T_A		–40	125	°C
	Junction, T_J			150	
	Storage, T_{stg}		–65	150	

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) V_{IN+} and $V_{IN–}$ can have a differential voltage of –26 V to +26 V; however, the voltage at these pins must not exceed the range of –0.3 V to +26 V.

7.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001(1)	±2500	V
		Charged-device model (CDM), per JEDEC specification JESD22-C101(2)	±1000	
		Machine model	±200	

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.
(2) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process.

7.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

	MIN	NOM	MAX	UNIT
Operating supply voltage	2.7		5.5	V
Operating temperature, T_A	–40		125	°C

7.4 Thermal Information

THERMAL METRIC(1)		INA3221 RGV (VQFN) 16 PINS	UNIT
$R_{\theta JA}$	Junction-to-ambient thermal resistance	36.5	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	42.7	°C/W
$R_{\theta JB}$	Junction-to-board thermal resistance	14.7	°C/W
ψ_{JT}	Junction-to-top characterization parameter	0.5	°C/W
ψ_{JB}	Junction-to-board characterization parameter	14.8	°C/W
$R_{\theta JC(bot)}$	Junction-to-case (bottom) thermal resistance	3.3	°C/W

(1) For more information about traditional and new thermal metrics, see the *Semiconductor and IC Package Thermal Metrics* application report (SPRA953).

7.5 Electrical Characteristics

at T_A = 25°C, V_S = 3.3 V, V_{IN+} = 12 V, V_{SHUNT} = (V_{IN+}) – (V_{IN-}) = 0 mV, and V_{BUS} = V_{IN-} = 12 V (unless otherwise noted)

PARAMETER			TEST CONDITIONS	MIN	TYP	MAX	UNIT
INPUT							
V_{SHUNT}	Shunt voltage input			–163.84		163.8	mV
V_{BUS}	Bus voltage input			0		26	V
CMR	Common-mode rejection		V_{IN+} = 0 V to +26 V	110	120		dB
V_{OS}	Shunt offset voltage, RTI(1)				±40	±80	µV
			T_A = –40°C to +125°C		0.1	0.5	µV/°C
PSRR			vs power supply, V_S = 2.7 V to 5.5 V		15		µV/V
V_{OS}	Bus offset voltage, RTI(1)				±8	±16	mV
			T_A = –40°C to +125°C			80	µV/°C
PSRR			vs power supply		0.5		mV/V
I_{IN+}	Input bias current at IN+				10		µA
I_{IN-}	Input bias current at IN–				10 ‖ 670		µA ‖ kΩ
	Input leakage(2)		(IN+ pin) + (IN– pin), power-down mode		0.1	0.5	µA
DC ACCURACY							
	ADC native resolution				13		Bits
	1-LSB step size		Shunt voltage		40		µV
			Bus voltage		8		mV
	Shunt voltage gain error				0.1%	0.25%	
			T_A = –40°C to +125°C		10	50	ppm/°C
	Bus voltage gain error				0.1%	0.25%	
			T_A = –40°C to +125°C		10	50	ppm/°C
DNL	Differential nonlinearity				±0.1		LSB
$t_{CONVERT}$	ADC conversion time		CT bit = 000		140	154	µs
			CT bit = 001		204	224	
			CT bit = 010		332	365	
			CT bit = 011		588	646	
			CT bit = 100		1.1	1.21	ms
			CT bit = 101		2.116	2.328	
			CT bit = 110		4.156	4.572	
			CT bit = 111		8.244	9.068	
SMBus							
	SMBus timeout(3)				28	35	ms
DIGITAL INPUT/OUTPUT							
C_I	Input capacitance				3		pF
	Leakage input current		0 V ≤ V_{IN} ≤ V_S		0.1	1	µA
V_{IH}	High-level input voltage			0.7 (V_S)		6	V
V_{IL}	Low-level input voltage			–0.5		0.3 (V_S)	V
V_{OL}	Low-level output voltage	SDA, critical, warning, PV	V_S > +2.7 V, I_{OL} = 3 mA	0		0.4	V
		TC	V_S > +2.7 V, I_{OL} = 1.2 mA	0		0.4	
V_{hys}	Hysteresis voltage				500		mV
POWER SUPPLY							
	Quiescent current				350	450	µA
			Power-down mode		0.5	2	
	Power-on reset threshold				2		V

(1) RTI = Referred-to-input.
(2) Input leakage is positive (current flows into the pin) for the conditions shown at the top of this table. Negative leakage currents can occur under different input conditions.
(3) SMBus timeouts in the INA3221 reset the interface whenever SCL is low for more than 28 ms.

7.6 Typical Characteristics

at T_A = 25°C, V_S = 3.3 V, V_{IN+} = 12 V, V_{SHUNT} = (V_{IN+}) – (V_{IN-}) = 0 mV, and V_{BUS} = V_{IN-} = 12 V (unless otherwise noted)

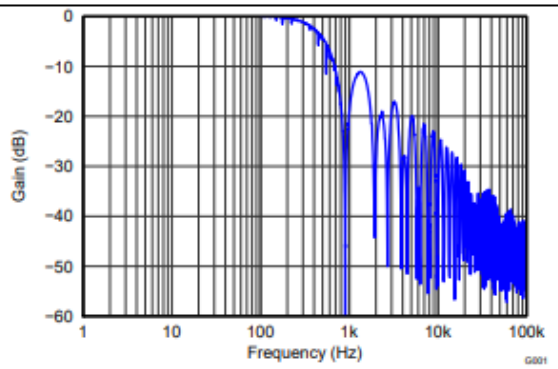


Figure 1. Frequency Response

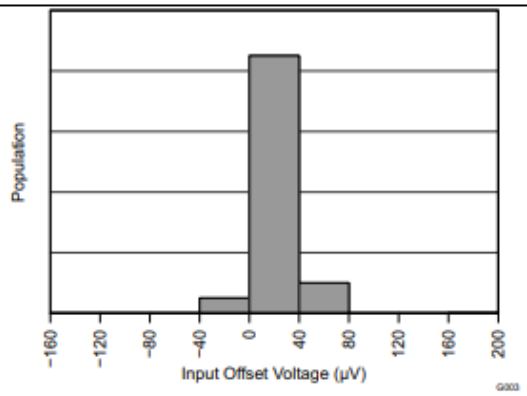


Figure 2. Shunt Input Offset Voltage Production Distribution

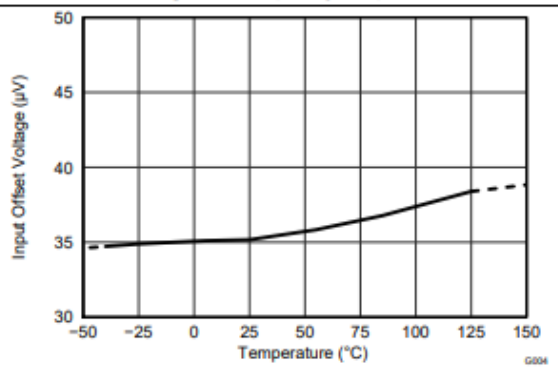


Figure 3. Shunt Input Offset Voltage vs Temperature

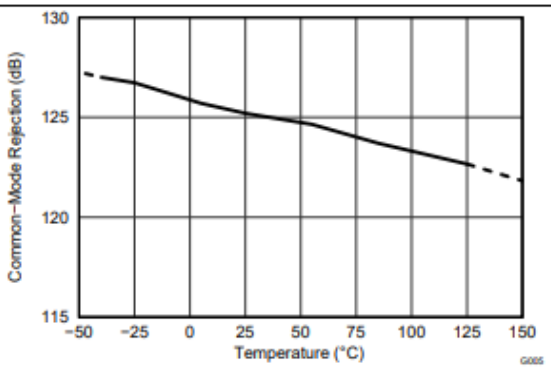


Figure 4. Shunt Input Common-Mode Rejection Ratio vs Temperature

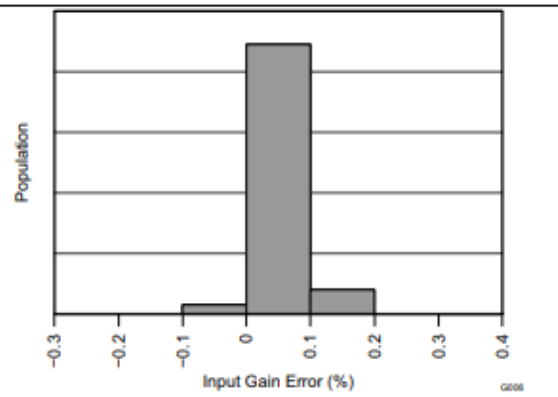


Figure 5. Shunt Input Gain Error Production Distribution

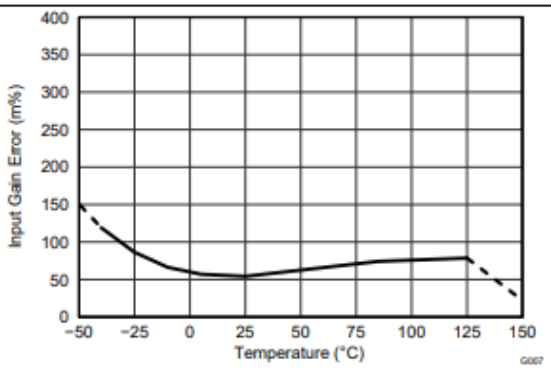


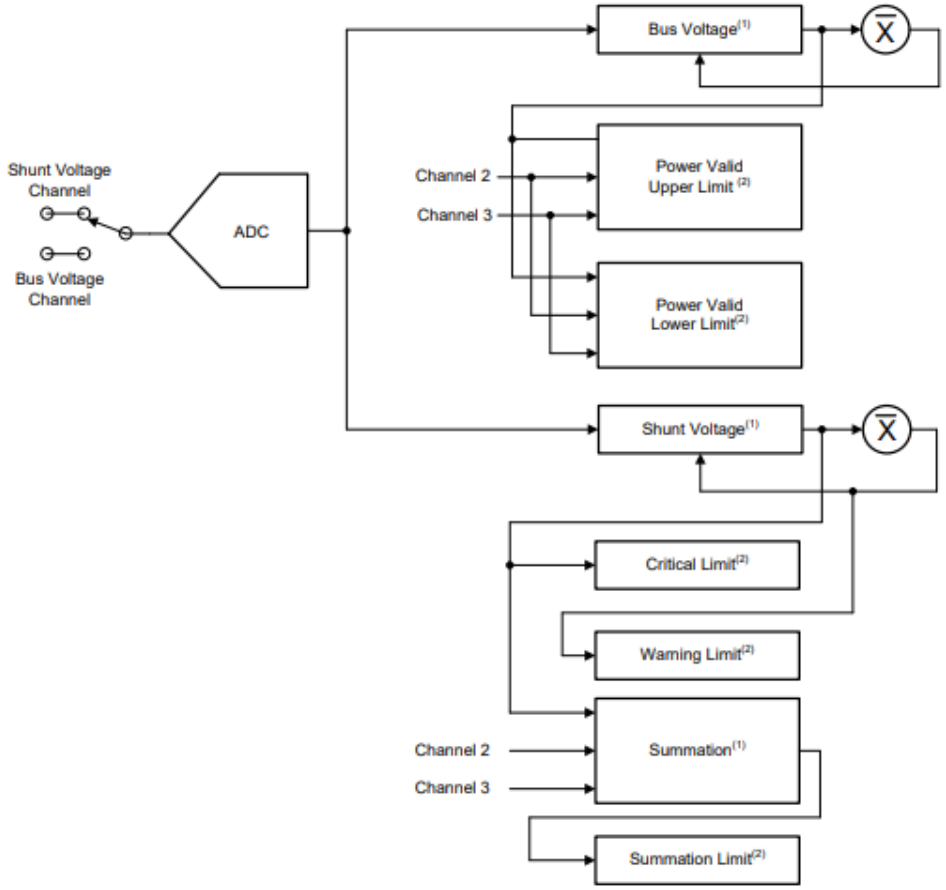
Figure 6. Shunt Input Gain Error vs Temperature

8 Detailed Description

8.1 Overview

The INA3221 is a current-shunt and bus voltage monitor that communicates over an I²C- and SMBus-compatible interface. The INA3221 provides digital shunt and bus voltage readings necessary for accurate decision making in precisely-controlled systems, and also monitors multiple rails to maintain compliance voltages. Programmable registers offer flexible configuration for measurement precision, and continuous versus single-shot operation. The *Register Maps* section provides details of the INA3221 registers, beginning with Table 3.

8.2 Functional Block Diagram



(1) Read-only.
(2) Read/write.

8.3 Feature Description

8.3.1 Basic ADC Functions

The INA3221 performs two measurements on up to three power supplies of interest. The voltage developed from the load current passing through a shunt resistor creates a shunt voltage that is measured between the IN+ and IN– pins. The device also internally measures the power-supply bus voltage at the IN– pin for each channel. The differential shunt voltage is measured with respect to the IN– pin, and the bus voltage is measured with respect to ground.

The INA3221 is typically powered by a separate power supply that ranges from 2.7 V to 5.5 V. The monitored supply buses range from 0 V to 26 V.

CAUTION

Based on the fixed 8-mV bus-voltage register LSB (for any channel), a full-scale register value results in 32.76 V. However, the actual voltage applied to the INA3221 input pins must not exceed 26 V.

There are no special power-supply sequencing considerations between the common-mode input ranges and the device power-supply voltage because they are independent of each other; therefore, the bus voltages can be present with the supply voltage off and vice versa.

The INA3221 takes two measurements for each channel: one for shunt voltage and one for bus voltage. Each measurement can be independently or sequentially measured, based on the mode setting (bits 2-0 in the Configuration register). When the INA3221 is in normal operating mode (that is, the MODE bits of the Configuration register are set to 111), the device continuously converts a shunt-voltage reading followed by a bus-voltage reading. This procedure converts one channel, and then continues to the shunt voltage reading of the next enabled channel, followed by the bus-voltage reading for that channel, and so on, until all enabled channels have been measured. The programmed Configuration register mode setting applies to all channels. Any channels that are not enabled are bypassed in the measurement sequence, regardless of mode setting.

The INA3221 has two operating modes, continuous and single-shot, that determine the internal ADC operation after these conversions complete. When the INA3221 is set to continuous mode (using the MODE bit settings), the device continues to cycle through all enabled channels until a new configuration setting is programmed.

The Configuration register MODE control bits also enable modes to be selected that convert only the shunt or bus voltage. This feature further allows the device to fit specific application requirements.

In single-shot (triggered) mode, setting any single-shot convert mode to the Configuration register (that is, the Configuration register MODE bits set to 001, 010, or 011) triggers a single-shot conversion. This action produces a single set of measurements for all enabled channels. To trigger another single-shot conversion, write to the Configuration register a second time, even if the mode does not change. When a single-shot conversion is initiated, all enabled channels are measured one time and then the device enters a power-down state. The INA3221 registers can be read at any time, even while in power-down. The data present in these registers are from the last completed conversion results for the corresponding register. The conversion ready flag bit (Mask/Enable register, CVRF bit) helps coordinate single-shot conversions, and is especially helpful during longer conversion time settings. The CVRF bit is set after all conversions are complete. The CVRF bit clears under the following conditions:

1. Writing to the Configuration register, except when configuring the MODE bits for power-down mode; or
2. Reading the Mask/Enable register.

In addition to the two operating modes (continuous and single-shot), the INA3221 also has a separate selectable power-down mode that reduces the quiescent current and turns off current into the INA3221 inputs. Power-down mode reduces the impact of supply drain when the device is not used. Full recovery from power-down mode requires 40 µs. The INA3221 registers can be written to and read from while the device is in power-down mode. The device remains in power-down mode until one of the active MODE settings are written to the Configuration register.

Feature Description (continued)

8.3.2 Alert Monitoring

The INA3221 allows programmable thresholds that make sure the intended application operates within the desired operating conditions. Multiple monitoring functions are available using four alert pins: Critical, Warning, PV (power valid), and TC (timing control). These alert pins are open-drain connections.

8.3.2.1 Critical Alert

The critical-alert feature monitors functions based on individual conversions of each shunt-voltage channel. The critical-alert limit feature compares the shunt-voltage conversion for each shunt-voltage channel to the value programmed into the corresponding limit register, in order to determine if the measured value exceeds the intended limit. Exceeding the programmed limit indicates that the current through the shunt resistor is too high.

At power-up, the default critical-alert limit value for each channel is set to the positive full-scale value, effectively disabling the alert. Program the corresponding limit registers at any time to begin monitoring for out-of-range conditions. The Critical alert pin pulls low if any channel measurement exceeds the limit present in the corresponding-channel critical-alert limit register. When the Critical alert pulls low, read the Mask/Enable register to determine which channel caused the critical alert flag indicator bit (CF1-3) to assert (= 1).

8.3.2.1.1 Summation Control Function

The INA3221 also allows the Critical alert pin to be controlled by the summation control function. This function adds the single shunt-voltage conversions for the desired channels (set by SCC1-3 in the Mask/Enable register) in order to compare the combined sum to the programmed limit.

The SCC bits either disable the summation control function or allow the summation control function to switch between including two or three channels in the Shunt-Voltage Sum register. The Shunt-Voltage Sum Limit register contains the programmed value that is compared to the value in the Shunt-Voltage Sum register in order to determine if the total summed limit is exceeded. If the shunt-voltage sum limit value is exceeded, the Critical alert pin pulls low. Either the summation alert flag indicator bit (SF) or the individual critical alert limit bits (CF1-3) in the Mask/Enable register determine the source of the alert when the Critical alert pin pulls low.

For the summation limit to have a meaningful value, use the same shunt-resistor value on all included channels. Unless equal shunt-resistor values are used for each channel, do not use this function to add the individual conversion values directly together in the Shunt-Voltage Sum register to report the total current.

8.3.2.2 Warning Alert

The warning alert monitors the averaged value of each shunt-voltage channel. The averaged value of each shunt-voltage channel is based on the number of averages set with the averaging mode bits (AVG1-3) in the Configuration register. The average value updates in the shunt-voltage output register each time there is a conversion on the corresponding channel. The device compares the averaged value to the value programmed in the corresponding-channel Warning Alert Limit register to determine if the averaged value has been exceeded, indicating whether the average current is too high. At power-up, the default warning-alert limit value for each channel is set to the positive full-scale value, effectively disabling the alert. The corresponding limit registers can be programmed at any time to begin monitoring for out-of-range conditions. The Warning alert pin pulls low if any channel measurements exceed the limit present in the corresponding-channel Warning Alert Limit register. When the Warning alert pin pulls low, read the Mask/Enable register in order to determine which channel warning alert flag indicator bit (WF1-3) is asserted (= 1).

OpenCV: Detección de objetos según el color

```
# Algoritmo que detecta los objetos segun el rango de color determinado
# Indica sus coordenadas (X,Y)

import sys
import cv2
import numpy as np

def read_cam():

   # Utiliza la camara incorporada de Jetson TX2
   cap = cv2.VideoCapture("nvcamerasrc ! video/x-raw(memory:NVMM), width=(int)1280,
height=(int)720,format=(string)I420, framerate=(fraction)30/1 ! nvvidconv flip-method=0 !
video/x-raw, format=(string)BGRx ! videoconvert ! video/x-raw, format=(string)BGR ! appsink")
   if cap.isOpened():
      windowName = "Detecccion de objetos"
      cv2.namedWindow(windowName, cv2.WINDOW_NORMAL)
      cv2.resizeWindow(windowName,1280,720)
      cv2.moveWindow(windowName,0,0)
      cv2.setWindowTitle(windowName,"Detecccion de objetos")
      showWindow=3
      font = cv2.FONT_HERSHEY_PLAIN

      while True:
         if cv2.getWindowProperty(windowName, 0) < 0: # Verifica si el usuario cerro la ventana
            # Esto fallara si el usuario cierra la ventana
            break;
         ret_val, frame = cap.read();

         # Convierte la imagen de formato RGB a HSV
         hsv=cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

         # Los colores detectados sera la gama azul
         hsv2=cv2.inRange(hsv,(80,50,5),(135,255,255),0.5)

         cv2.getStructuringElement(cv2.MORPH_RECT,(10,10))
         np.array([[1,1,1,1,1,1,1,1,1,1],
                   [1,1,1,1,1,1,1,1,1,1],
                   [1,1,1,1,1,1,1,1,1,1],
                   [1,1,1,1,1,1,1,1,1,1],
                   [1,1,1,1,1,1,1,1,1,1],
                   [1,1,1,1,1,1,1,1,1,1],
                   [1,1,1,1,1,1,1,1,1,1],
                   [1,1,1,1,1,1,1,1,1,1],
                   [1,1,1,1,1,1,1,1,1,1],
                   [1,1,1,1,1,1,1,1,1,1]],dtype=np.uint8)

         kernel=np.ones((10,10),np.uint8)
         erosion=cv2.erode(hsv2,kernel,iterations=1)
```

```
        dilatado=cv2.dilate(erosion,kernel,iterations=1)

        # Busca el contorno y los datos los guarda en el vector=contorno
        imagen, contorno, jerarquia =cv2.findContours(dilatado, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

        # Colorea la parte del objeto detectada de ese color
        dibujo=cv2.drawContours(dilatado, contorno, -1,(160,160,160),cv2.FILLED)

        if showWindow == 3:

         # Obtiene las coordenadas y dibuja el cuadrado que lo enmarca
         for c in contorno:
            x,y,w,h=cv2.boundingRect(c)

            rectangulo=cv2.rectangle(dibujo,(x,y),(x+w,y+h),(255,255,255),5,8,0)

            CentroX=x
            CentroY=y

            # Transforma de entero a cadena
            EjeX=str(CentroX)
            EjeY=str(CentroY)

            # Muestra por pantalla las coordenadas
            cv2.putText(dibujo,EjeX,(20,40),font, 3,(255,255,255),1, cv2.LINE_AA)
            cv2.putText(dibujo,EjeY,(120,40),font, 3, (255,255,255),1, cv2.LINE_AA)

            camara=cv2.resize(frame,(420,420))
            cv2.imshow("Camara",camara)

        cv2.imshow(windowName,dibujo)
        key=cv2.waitKey(10)
        if key == 27: # Verifica la tecla ESC
           cv2.destroyAllWindows()
           break ;

    else:
     print "camera open failed"

if __name__ == '__main__':
    read_cam()
```

OpenCV: Detección de circuferencias

```
# Algoritmo que detecta las circuferencias y las dibuja
# Indica su radio y si este es superior a un rango establecido

import sys
import cv2
import numpy as np

def read_cam():

   # Utiliza la camara incorporada de Jetson TX2
   cap = cv2.VideoCapture("nvcamerasrc ! video/x-raw(memory:NVMM), width=(int)1280,
height=(int)720,format=(string)I420, framerate=(fraction)30/1 ! nvvidconv flip-method=0 !
video/x-raw, format=(string)BGRx ! videoconvert ! video/x-raw, format=(string)BGR ! appsink")
   if cap.isOpened():
      windowName = "Detecccion de circuferencias"
      cv2.namedWindow(windowName, cv2.WINDOW_NORMAL)
      cv2.resizeWindow(windowName,1280,720)
      cv2.moveWindow(windowName,0,0)
      cv2.setWindowTitle(windowName,"Detecccion de circuferencias")
      showWindow=3
      font = cv2.FONT_HERSHEY_PLAIN

      while True:
         if cv2.getWindowProperty(windowName, 0) < 0: # Verifica si el usuario cerro la ventana
            # Esto fallara si el usuario cierra la ventana
            break;
         ret_val, frame = cap.read();

         # Convierte la imagen de formato RGB a GRIS
         gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

         if showWindow == 3:

            # Busca si hay circunferencias y los datos los guarda en el vector=datos

datos=cv2.HoughCircles(gray,cv2.HOUGH_GRADIENT,1,1,param1=100,param2=100,minRadius
=0,maxRadius=100)

            if datos == None :
               continue
            print datos

            # Dibuja el circulo que lo enmarca
            for i in datos[0,:]:

               dibujo=cv2.circle(gray,(i[0],i[1]),i[2],(0,255,0),2)

               # Se determina el rango
```

```
            if i[2]<50:

               rango="Es menor a 100"

            else:

               rango="Es mayor a 100"

            Radio=i[2]

            # Transforma de entero a cadena
            dato=str(Radio)

         # Muestra por pantalla el valor del radio y si supera el rango
         cv2.putText(gray,dato, (10,40), font,3, (255,255,255), 1, cv2.LINE_AA)
         cv2.putText(gray, rango, (10,100), font, 3, (255,255,255), 1, cv2.LINE_AA)
      cv2.imshow(windowName,gray)
      key=cv2.waitKey(10)
      if key == 27: # Check for ESC key
         cv2.destroyAllWindows()
         break ;

   else:
    print "camera open failed"



if __name__ == '__main__':
   read_cam()
```

Pascal: 20 Clases de objetos

Esta lista de objetos se ha tomado como referencia de (Redmon) [84]

- Autobús
- Avión
- Barco
- Bicicleta
- Botella
- Caballo
- Coche
- Gato
- Mesa de comedor
- Moto
- Oveja
- Pájaro
- Perro
- Persona
- Planta en maceta
- Silla
- Sofá
- Tren
- Tv / Monitor
- Vaca

COCO: 80 Clases de objetos

Esta lista de objetos se ha tomado del archivo "*coco.names*".

- Autobús
- Avión
- Banco
- Barco
- Bate de béisbol
- Baño
- Bicicleta
- Boca de incendio
- Bolso
- Botella
- Brócoli
- Caballo
- Cama
- Camión
- Cebra
- Cepillo de dientes
- Coche
- Comedor
- Cometa
- Copa de vino
- Corbata
- Cuchillo
- Cuchara
- Cuenco
- Elefante
- Emparedado
- Esquís
- Florero
- Frisbee
- Gato
- Guante de béisbol
- Horno
- Jirafa
- Lavabo
- Libro
- Maleta
- Manzana
- Microonda
- Mochila
- Moto
- Naranja
- Ordenador portátil
- Oso
- Oso de peluche
- Oveja
- Pájaro
- Pancho
- Paraguas
- Parquímetro
- Pastel
- Patineta
- Pelota deportiva
- Perro
- Persona
- Pizza
- Planta en maceta
- Plátano
- Raqueta de tenis
- Ratón
- Refrigerador
- Remoto
- Reloj
- Rosquilla
- Secadora de pelo
- Semáforo
- Señal de stop
- Silla
- Snowboard
- Sofá
- Tabla de surf
- Teclado
- Teléfono móvil
- Tenedor
- Tijeras
- Tostadora
- Tren
- Tv / Monitor
- Vaca
- Vaso
- Zanahoria

Cálculos para determinar la Precisión y Recall

Estos son los cálculos realizados en el ejemplo mostrado en el apartado 5.5.1 YOLOv1. En él se utilizaron las siguientes ecuaciones.

$$P = \frac{TP}{(TP + FP)} \qquad R = \frac{TP}{(TP + FN)}$$

Valores correspondientes a la primera posición de las predicciones del ranking.

$$P_1 = \frac{1}{1} = 1 \qquad R_1 = \frac{1}{5} = 0.2$$

Valores correspondientes a la segunda posición de las predicciones del ranking.

$$P_2 = \frac{2}{2} = 1 \qquad R_2 = \frac{2}{5} = 0.4$$

Valores correspondientes a la tercera posición de las predicciones del ranking.

$$P_3 = \frac{3}{3} = 1 \qquad R_3 = \frac{3}{5} = 0.6$$

Valores correspondientes a la cuarta posición de las predicciones del ranking.

$$P_4 = \frac{3}{3+1} = 0.75 \qquad R_4 = \frac{3}{5} = 0.6$$

Valores correspondientes a la quinta posición de las predicciones del ranking.

$$P_5 = \frac{3}{3+2} = 0.6 \qquad R_5 = \frac{3}{5} = 0.6$$

Valores correspondientes a la sexta posición de las predicciones del ranking.

$$P_6 = \frac{4}{4+2} = 0.67 \qquad R_6 = \frac{4}{5} = 0.8$$

Valores correspondientes a la séptima posición de las predicciones del ranking.

$$P_7 = \frac{4}{4+3} = 0.58 \qquad R_7 = \frac{4}{5} = 0.8$$

Valores correspondientes a la octava posición de las predicciones del ranking.

$$P_8 = \frac{4}{4+4} = 0.5 \qquad R_8 = \frac{4}{5} = 0.8$$

Valores correspondientes a la novena posición de las predicciones del ranking.

$$P_9 = \frac{5}{5+4} = 0.51 \qquad R_9 = \frac{5}{5} = 1$$

Valores correspondientes a la décima posición de las predicciones del ranking.

$$P_{10} = \frac{5}{5+5} = 0.5 \qquad R_{10} = \frac{5}{5} = 1$$

Comprobación mediante ejemplo

El estudio que se muestra a continuación muestra los diferentes resultados que se obtienen al utilizar las tres versiones de YOLO y al trabajar con diferentes tamaños de imágenes. La información de este estudio corresponde a (Evaluating State-of-the-art Object Detector on Challenging Traffic Light Data, n.d.) [27]

En él se plantea el entrenamiento de tres bases de datos que son:

Base de datos	Resolución	Nº Clases	Clases
LARA	640x480	4	Verde, naranja, rojo y ambiguo
LISA-Train	1280x960	6	Ir, ir a la izquierda, advertencia, advertencia a la izquierda, stop, stop a la izquierda
LISA-Seq2	1280x960	6	Ir, avanzar, ir a la izquierda, advertencia, stop, stop a la izquierda

Con estos datos se realizaron diferentes entrenamientos en los que en algunos se combinaron las bases de datos y se habilitó el comando *Random*. Estas son las diferentes combinaciones que se presentan:

- YOLO_V1_0: Utilizo la base de datos LISA-Train.
- YOLO_V1_1: Utilizo la base de datos LISA-Train y el comando *Random*.
- YOLO_V2_0: Utilizo las bases de datos LISA-Train y LARA.
- YOLO_V2_1: Utilizo las bases de datos LISA-Train, LARA y el comando *Random*.
- YOLO_V3_0: Utilizo las bases de datos LISA-Train y LISA_Seq2.
- YOLO_V3_1: Utilizo las bases de datos LISA-Train, LISA_Seq2 y el comando *Random*.

Las siguientes gráficas muestran la relación Precisión-*Recall* de las predicciones obtenidas en función de las bases de datos anteriores. Se muestran dos, ya que se realizaron con dos resoluciones de imágenes, una de (416x416) y la otra de (672x672).

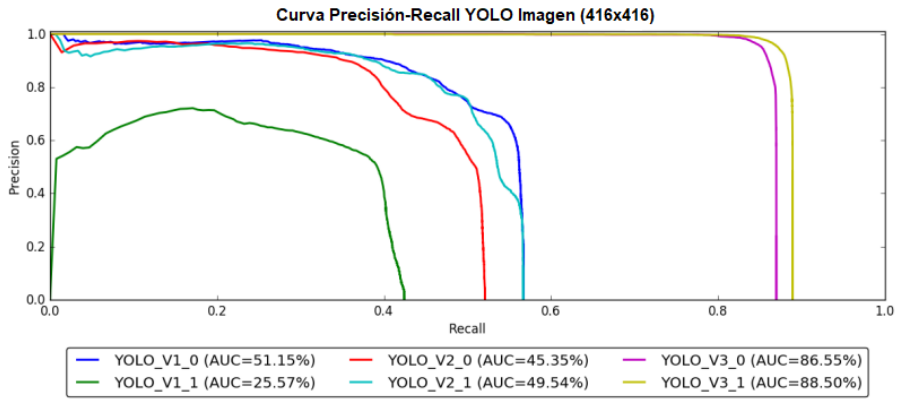


Figura 173: Curva Precisión-Recall (416x416)

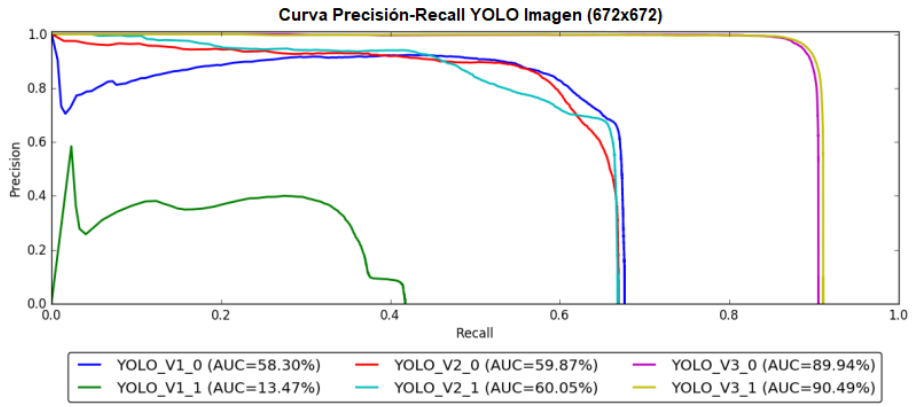


Figura 174: Curva Precisión-Recall (672x672)

Donde el parámetro (AUC) hace referencia al área que se encuentra debajo de la curva generada. Es decir, es la precisión media (AP) obtenida de cada entrenamiento realizado.

En los gráficos se observa que mejores resultados se obtienen con la imagen de mayor resolución (672x672). Esto se debe que cuando las capas de la red neuronal reducen el tamaño de la imagen los objetos no se distorsionan tanto. Además, se observa que mejores resultados se obtienen si el comando *Random* está habilitado. Ya que el cambiar el tamaño de las imágenes de manera aleatoria cada diez lotes, hace que la red neuronal en el entrenamiento capte más información, obteniendo de este modo mejores predicciones. La excepción se encuentra en YOLO_V1_1 el cual al aumentar la resolución de la imagen obtiene peores resultados, esto se puede deber a que el número de datos que se utilizan no son los suficientes. Por lo que al aumentar la resolución de la imagen y estar activo el comando *Random* hace que el sistema se vuelva inestable.

Las dos imágenes siguientes muestran las gráficas *Recall*-Iteraciones obtenidas al utilizar las dos resoluciones.

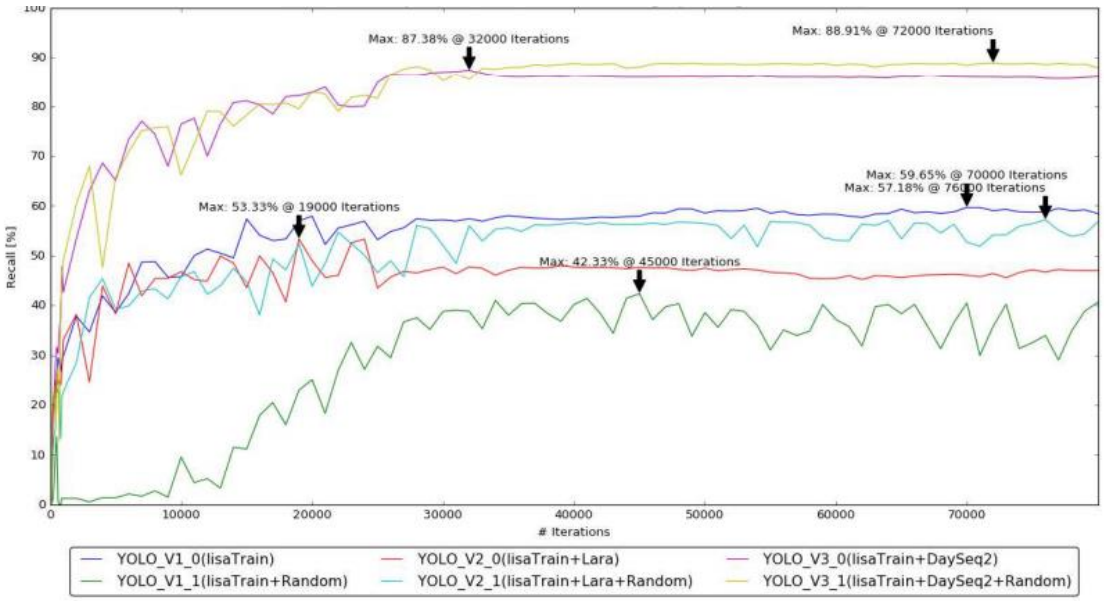


Figura 175: Recall-Iteraciones (416x416)

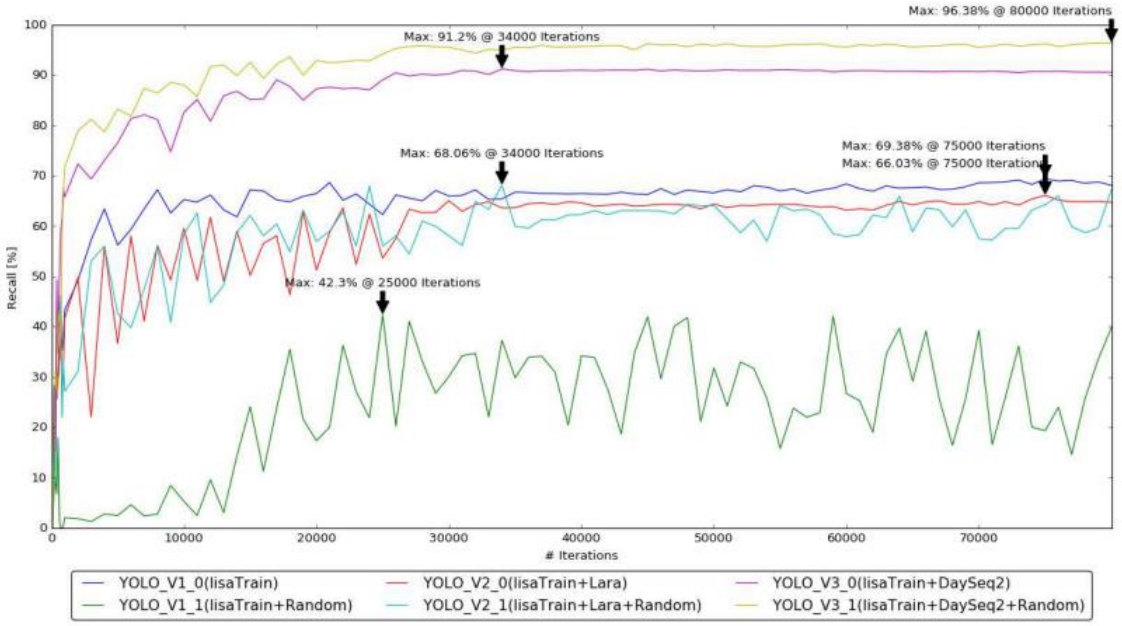


Figura 176: Recall-Iteraciones (672x672)

En esta gráfica se indica como de favorable son las predicciones que se han obtenido con respecto las iteraciones que se han realizado.

Se observa que al igual que el caso anterior se obtiene mejores resultados cuanto mayor es la resolución de la imagen, un caso llamativo es el que se observa en los datos de YOLO_V1_1 en los que con una resolución de (416x416) y 45000 iteraciones se obtiene un 42.33%. Mientras que si la resolución es de (672x672) se requiere de 25000 iteraciones para alcanzar el valor máximo de 42.3%. Realizando 20000 iteraciones menos para conseguir un valor muy similar, con sólo una pérdida del 0.03.

En algunos de los otros casos no vale la pena aumentar el tamaño o realizarlo con el comando *Random,* ya que la diferencia es mínima. Ejemplo de ello es:

El valor de (*Recall %)* máximo que obtiene YOLO_V3_0 es de 87.38 % en la iteración 32000, mientras que con Random y con la misma resolución el valor es de 88.91% pero con 72000 iteraciones. Necesitando de este modo realizar 40000 iteraciones más para conseguir una mejora del 1.53%.

El mejor valor obtenido corresponde a YOLO_V3 para ambas resoluciones, en el que el valor de *Recall* es alto y requiere de un valor intermedio de iteraciones.

Pliego de condiciones

Tabla de contenido

Condiciones Hardware

Para poder realizar el TFG los elementos mínimos que se necesitan son:

- Tarjeta NVIDIA Jetson TX2.
- Una pantalla con entrada HDMI.
- Los periféricos teclado y ratón con conexión USB, para poder interactuar con el sistema.
- Internet con una buena conexión Wifi o realizarla mediante un cable tipo RJ-45.

Para realizar las pruebas es necesario tener espacio libre en la memoria, lo suficiente como para poder descargar los algoritmos o realizar el entrenamiento de las redes neuronales. En este caso se trabajó con 4 GB libres, una alternativa es trabajar con un disco duro, el cual se puede conectar mediante conexión USB o SATA.

Otra alternativa es incorporarle otras cámaras, ya que es posible la conexión de hasta seis cámaras. En este caso se realizó todo mediante la cámara incorporada en la tarjeta Jetson TX2.

Condiciones térmicas

En la siguiente tabla se muestran las especificaciones térmicas del sistema.

Parámetro	Valor
Temperatura máxima de funcionamiento TTP	80 ° C
Límite de temperatura de funcionamiento recomendada	T.cpu = 95.5 ° C
	T.gpu = 93.5 ° C
Límite máximo de temperatura de funcionamiento	T.cpu = 101 ° C
	T.gpu = 101 ° C

El ventilador situado en la parte superior del disipador de calor empieza a funcionar cuando se alcanzan los 51 °C, y deja de funcionar cuando se encuentra a 35 °C o por debajo de esta temperatura.

Condiciones de alimentación

La fuente de alimentación que se le conecta a la SBC Jetson TX2 es de 19 V y 4.7 A. Una de las principales características de este dispositivo es que su módulo consume sólo 15 W, siendo su potencia total de 90 W.

En el caso de que se le conecte una batería los valores más comunes son 2.5 V y 3.5 V.

Condiciones de conexión

Se requiere de un conector HDMI A para conectar la pantalla al dispositivo y poder observar los resultados. Como ya se ha dicho anteriormente se necesita un teclado y ratón para poder trabajar con la tarjeta, y en el caso de que se utilice un cable RJ-45 la longitud máxima que alcanza es de 100 m. En el caso de que se necesite más distancia, se ha de colocar repetidores.

Condiciones Software

Los requisitos software necesarios para su desarrollo fueron la implementación del SO Linux y Ubuntu 16.04, además de la descarga del Jetpack 3.1. Este último incorpora un conjunto de librerías y programas como Opencv, Tensort o OpenGL

Para la ejecución de los respectivos algoritmos utilizados en el TFG es necesario su descarga e implementación en la tarjeta. Al igual que para el entrenamiento de YOLOv3 en el que es necesario la descarga de las respectivas imágenes para el aprendizaje y la creación de los archivos necesarios.

Sistemas Operativos

- SO Linux 4.4.38: Compatible con C, C++, ensamblador, Perl, Python y con otros lenguajes Shell scripting.
- Ubuntu 16.04 LTS: Es compatible su instalación en un mismo equipo con otro sistema operativo como Windows.

Jetpack 3.1

A continuación, se muestra el conjunto de herramientas y demos, así como las versiones que soporta y es compatible el sistema.

- Tegra R28.1
- Cámara V4L2/API de códec
- CUDA Toolkit 8.0.82
- TensorRT 2.1 GA
- VisionWorks 1.6
- OpenGL
  - OpenGL 4.6
  - OpenGL 4.5
  - OpenGL ES 3.2
  - OpenGL ES 3.1
  - EGL 1.5

- OpenCV 2.4.13
- Tegra System Profiler 3.8 (TSP)
- API Tegra Multimedia 27.1
- CuDNN 6.0 (Cuda Deep Neural Network)
- GStreamer 1.8.2

Otros

- Jetson-Inference
- Jetson-Reinforcement
- YOLOv3: La velocidad que se alcanza cuando la ejecución se realiza en una Jetson TX2 es de 3 FPS.
- OpenCV 3.4.0, ya que la versión 3.4.1 no es compatible con YOLOv3.

Valoración Económica

Tabla de contenido

Recursos Hardware

La realización de este trabajo se basa en el análisis y uso de la tarjeta Jetson TX2, la cual salió al mercado en 2017. Para llevar a cabo la experimentación y comprobación de los ejemplos se requiere de una pantalla monitor para ver los resultados y de los dispositivos teclado y ratón. En las pruebas se utilizó la cámara que lleva incorporada la tarjeta Jetson TX2, por lo que no fue necesario adquirir otra.

Material	Unidades	Importe (€)	Subtotal (€)
Tarjeta Jetson TX2	1	600 €	600 €
Pantalla monitor	1	110 €	710 €
Teclado	1	25 €	735 €
Ratón	1	40 €	745 €
Cámara	1	-	745 €
Total			745 €
Asciende el citado apartado de Recursos Hardware a la cantidad de SETECIENTOS CUARENTA Y CINCO EUROS			

Recursos Software

Los algoritmos y demos que se utilizaron para llevar a cabo la simulación de los ejemplos, la realización de los programas OpenCV y el reentrenamiento de la red neuronal son todos de código abierto por lo que no hay que pagar sus licencias para conseguirlas.

Software		Versión	Precio
Jetpack	TensorRT	3.1	Gratuito
	VisionWorks		
	Cámara V4L2		
	Tegra System Profiles		
	CuDNN		
	GStreamer		
	OpenGL		
CUDA	Partículas	8.0	Gratuito
	Fluido		
	Random Fog		
Jetson-Inference	ImageNet	-	Gratuito
	DetecNet		
	SegNet		
Jetson-Reinforcement	Cartpole	-	Gratuito
	Lunar Lander		
	Fruta		
	Brazo Robótico		
	Rover Navigation		
OpenCV	Coversión RGB	3.4.0	Gratuito
	Detector de objetos - color		
	Detector de circuferencias		
YOLO	Fotografía	3	Gratuito
	Vídeos		
	Cámara		
Total		0 €	
Asciende el citado apartado de Recursos Software a la cantidad de CERO EUROS			

Horas invertidas

A continuación, se muestra una aproximación de las horas invertidas en realizar este trabajo.

Tarea Global	Tarea realizada	Horas	Importe (€)	Subtotal (€)
Documentación	Búsqueda de información	250 horas	20 €/h	5000 €
	Búsqueda de conceptos desconocidos			
	Desarrollo del documento			
Experimentación	Descarga de demos	70 horas	20 €/h	1400 €
	Ejecución de algoritmos			
	Búsqueda de solución			
	Realización de los dos programas			
Reentrenamiento	Entrenamiento de botellas	86 horas	20 €/h	1720 €
	Entrenamiento de latas			
	Entrenamiento de logos			
	Realización de pruevas			
Total		406 horas	20 €/h	8120 €
Asciende el citado apartado de horas invertidas a la cantidad de OCHO MIL CIENTO VEINTE EUROS				

Coste Total

El coste total de la realización de este trabajo con IVA y beneficio industrial incluidos asciende al valor de 11258.55 €.

Concepto	Importe (€)	Subtotal (€)
Recursos Hardware	745 €	745 €
Recursos Software	0 €	745 €
Horas invertidas	8120 €	8865 €
Total	8865 €	8865 €
Total con IVA (21 %)	1861.65 €	10726.65 €
Total con Beneficio Industrial (6 %)	531.9 €	11258.55 €
Asciende el trabajo desarrollado a la cantidad total de ONCE MIL DOSCIENTOS CINCUENTA Y OCHO EUROS CON CINCUENTA Y CINCO CÉNTIMOS		

Logroño, 6 de septiembre de 2018

Nota:

- Se han utilizado precios de agosto 2018. En el supuesto de que se desee utilizar el TFG transcurridos 18 meses se recomienda actualización de precios.
- Las versiones del software utilizadas corresponden a las disponibles en agosto de 2018. Se ha de tener en cuenta posibles actualizaciones y nuevas versiones.